
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ**
**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**


# MĚŘENÍ RYCHLOSTI AUTOMOBILŮ Z DOHLÉDOVÉ KAMERY

**SPEED MEASUREMENT OF VEHICLES FROM SURVEILLANCE CAMERA**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** **Bc. SAMUEL JAKLOVSKÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. JAKUB SOCHOR**
**SUPERVISOR**



**BRNO 2018**

*Zadání diplomové práce/19643/2017/xjaklo02*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

# Zadání diplomové práce

Řešitel: **Jaklovský Samuel, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Měření rychlosti automobilů z dohlédové kamery**
**Speed Measurement of Vehicles from Surveillance Camera**
Kategorie: Zpracování obrazu

Pokyny:

1. Nastudujte algoritmy pro automatickou kalibraci dopravní dohledové kamery.
2. Nastudujte algoritmy pro měření rychlosti s pomocí jedné zkalibrované dohledové kamery.
3. Navrhněte systém pro kalibraci a měření rychlosti projíždějících automobilů před dohledovou kamerou.
4. Navržený systém implementujte.
5. Proveďte experimenty se systémem a vyhodnoďte jeho přesnost.
6. Vytvořte video pro prezentaci projektu.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 a 4.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Sochor Jakub, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
*vedoucí ústavu*

# Abstrakt

Táto diplomová práca sa zaoberá plne automatickou kalibráciou dopravnej dohľadovej kamery, ktorá je následne použitá na meranie rýchlosti prechádzajúcich vozidiel. Práca obsahuje a popisuje teoretické informácie a algoritmy týkajúce sa tejto problematiky. Na ich základe bol postavený komplexný návrh systému pre automatickú kalibráciu a meranie rýchlosti. Navrhnutý systém bol úspešne naimplementovaný. Implementovaný systém je optimalizovaný tak, aby pre automatickú kalibráciu kamery musel spracovať čo najmenší úsek vstupného videa. Kalibračné parametre sú tak získané po spracovaní iba dva a pol minúty vstupného videa. Presnosť implementovaného systému bola vyhodnotená na datasete BrnoCompSpeed. Chyba pri meraní rýchlosti pri použití systému automatickej kalibrácie predstavuje 8,15 km/h. Chyba je spôsobená hlavne nepresným získavaním mierky, pri jej nahradení manuálne získaným údajom sa nepresnosť zníži na 2,45 km/h. Samotný systém merania rýchlosti vykazuje chybu len 1,62 km/h (vyhodnotené použitím manuálne získanými kalibračnými parametrami).



# Abstract

This master's thesis is focused on fully automatic calibration of traffic surveillance camera, which is used for speed measurement of passing vehicles. Thesis contains and describes theoretical information and algorithms related to this issue. Based on this information and algorithms, a comprehensive system design for automatic calibration and speed measurement was built. The proposed system has been successfully implemented. The implemented system is optimized to process the smallest portion of the video input for the automatic calibration of the camera. Calibration parameters are obtained after processing only two and half minutes of input video. The accuracy of the implemented system was evaluated on the dataset BrnoCompSpeed. The speed measurement error using the automatic calibration system is 8.15 km/h. The error is mainly caused by inaccurate scale acquisition, and when it is replaced by manually obtained scale, the error is reduced to 2.45 km/h. The speed measuring system itself has an error of only 1.62 km/h (evaluated using manual calibration parameters).



# Kľúčové slová

Meranie rýchlosti, dohľadová kamera, sledovanie dopravy, kalibrácia kamery, plne automatická kalibrácia, detekcia úbežníku, detekcia vozidiel, klasifikácia vozidiel

# Keywords

Speed measurement, surveillance camera, traffic surveillance, camera calibration, fully automatic calibration, vanishing point detection, vehicle detection, vehicle classification


# Citácia

JAKLOVSKÝ, Samuel. *Měření rychlosti automobilů z dohlédové kamery.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jakub Sochor

# Měření rychlosti automobilů z dohlédové kamery

## Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Jakuba Sochora. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Samuel Jaklovský
21. mája 2018


## Poďakovanie

Chcel by som sa poďakovať svojmu vedúcemu práce Ing. Jakubovi Sochorovi za odbornú pomoc a usmernenie pri písaní tejto práce, za jeho čas, cenné rady a informácie a v neposlednom rade za ochotu.

Špeciálne poďakovanie patrí mojej manželke, rodičom a mojim najbližším za ich podporu a pomoc, vďaka ktorej som mohol túto prácu dokončiť.

# Obsah


**1 Úvod** **5**

**2 Existujúce riešenia** **6**
2.1 Kalibrácia dopravnej kamery . . . . . 6
2.1.1 Manuálna kalibrácia . . . . . 7
2.1.2 Automatická kalibrácia . . . . . 9
2.1.3 Princíp kalibrácie dopravnej kamery . . . . . 9
2.2 Meranie rýchlosti prechádzajúcich vozidiel . . . . . 11

**3 Použité metódy** **13**
3.1 Detekcia úbežníkov . . . . . 13
3.1.1 Diamond Space akumulátor . . . . . 13
3.2 Konvolučné neurónové siete . . . . . 15
3.3 Detekcia vozidiel . . . . . 16
3.3.1 SSD . . . . . 16
3.3.2 Faster R-CNN . . . . . 17
3.3.3 Porovnanie SSD a Faster R-CNN . . . . . 17
3.4 Klasifikácia vozidiel . . . . . 18
3.4.1 Hlboké residuálne učenie . . . . . 19
3.4.2 Residuálne siete . . . . . 19

**4 Návrh riešenia** **21**
4.1 Detekcia vozidiel . . . . . 22
4.2 Automatická kalibrácia . . . . . 23
4.2.1 Získanie prvého úbežníka . . . . . 25
4.2.2 Získanie druhého úbežníka . . . . . 27
4.2.3 Klasifikátor vozidiel . . . . . 29
4.2.4 Dopočítanie mierky . . . . . 30
4.3 Meranie rýchlosti . . . . . 32
4.3.1 Trackovanie vozidiel . . . . . 33
4.3.2 Dopočítanie rýchlosti . . . . . 33

**5 Implementácia** **36**

**6 Testovanie a vyhodnotenie** **38**
6.1 Testovacia sada . . . . . 38
6.2 Parametre testovania . . . . . 39
6.3 Výsledky testovania . . . . . 40

6.3.1 Testované a porovnávané verzie . . . . . 40
6.3.2 Vyhodnotenie výsledkov . . . . . 41

**7 Záver** **45**

**Literatúra** **46**

**A Obsah priloženého pamäťového média** **49**

**B Spustenie** **50**

**C Systémové požiadavky** **53**

# Zoznam obrázkov


2.1 Vzdialenosť dvoch bodov v obraze premietnutá na rovinu cesty. . . . . 6
2.2 Vyznačenie vodiacich čiar použité pre kalibráciu kamery. (Obr. prevzatý z [26]) 7
2.3 Nákres požadovaných parametrov pre riešenie od Paia a kol. [19]. . . . . 8
2.4 Scéna s rovnostranným trojuholníkom vyznačeným na zemi. (Obr. prevzatý z [7]) . . . . . 9
2.5 Ilustrácia vodiacich čiar a jazdných pruhov zbiehajúcich sa v úbežníku ( *VP*). (Obr. prevzatý z [6]) . . . . . 10
2.6 Mapa aktivity a získané hranice jazdných pruhov (nad snímkami z rozdielnych kamier). (Obr. prevzatý z [21]) . . . . . 11
2.7 Sledovanie zistených profilov (snímka úplne vľavo) v troch po sebe nasledujúcich snímkach. (Obr. prevzatý z [12]) . . . . . 12

3.1 Dve kaskádové transformácie bodov a priamok do paralelných súradníc. (Obr. prevzatý z [9]) . . . . . 14
3.2 Reprezentácia priamky v rozdielnych transformáciách. Iba tmavé trojuholníkové podpriestory sú súčasťou výsledného *diamond space*. (Obr. prevzatý z [9]) . . . . . 14
3.3 Kvadranty paralelných priestorov (vpravo) zodpovedajúce kvadrantom pôvodného nekonečného Karteziánskeho priestoru (vľavo). (Obr. prevzatý z [9]) 15
3.4 Príklad siete s mnohými konvolučnými vrstvami. (Obr. prevzatý z [1]) . . . 16
3.5 Diagram meta-architektúry detektora *SSD* (*Single Shot Detector*). (Obr. prevzatý z [15]) . . . . . 16
3.6 Diagram meta-architektúry detektora *Faster R-CNN* (*Regions with Convolutional Neural Network*). (Obr. prevzatý z [15]) . . . . . 17
3.7 Porovnanie presnosti jednotlivých architektúr a spôsobu implementácie extraktora čŕt voči času potrebnému na výpočet. (Obr. prevzatý z [15]) . . . . 18
3.8 Príklad detekcie osôb (*person*) a šarkanov (*kite*) najrýchlejšiou verziou *SSD* (vľavo) a najpresnejšou verziou *Faster R-CNN* (vpravo). (Obr. prevzatý z [15]) 18
3.9 Stavebný blok *residuálneho* učenia. (Obr. prevzatý z [13]) . . . . . 19
3.10 Vľavo: Stavebný blok siete *ResNet-34*. Vpravo: *Bottleneck* stavebný blok siete *ResNet-50*. (Obr. prevzatý z [13]) . . . . . 20

4.1 Hlavné bloky navrhovaného systému . . . . . 21
4.2 Hlavné bloky navrhovaného systému s vyčleneným blokom pre detekciu vozidiel. . . . . 22
4.3 Jednoduchá schéma komponentu detektora vozidiel. . . . . 22
4.4 Ukážka boxov ohraničujúcich detegované vozidlá vykreslených do obrazu. . 23
4.5 Schéma podsystému zabezpečujúceho automatickú kalibráciu. . . . . 24
4.6 Model a súradný systém kamery. (Obr. prevzatý z [22]) . . . . . 24

4.7 Schéma podsystému zabezpečujúceho určenie prvého úbežníka. . . . . 26
4.8 Vykreslené detegované kľúčové body vhodné na sledovanie sa nachádzajú vľavo. Vykreslené úsečky reprezentujúce pohyb týchto kľúčových bodov voči predchádzajúcim snímkam sa nachádzajú vpravo. . . . . 26
4.9 Demonštratívny grafický výstup komponentu s vykreslenými šípkami smerujúcimi k prvému úbežníku **u**. . . . . 27
4.10 Schéma podsystému zabezpečujúceho určenie druhého úbežníka. . . . . 27
4.11 Vizualizácia detekcie a filtrácie hrán. Zľava doprava – *Seedpoints* ako lokálne maximá veľkostí gradientu obrazu; Detegované hrany, na ktorých dané *seedpoints* ležia; *Seedpoints* ležiace na hranách, ktoré ostali po filtrácii; Hrany, ktoré ostali po filtrácii na základe uhlu a kvality. . . . . 28
4.12 Demonštratívny grafický výstup komponentu s vykreslenými šípkami smerujúcimi k prvému úbežníku **u** (červená farba), druhému úbežníku **v** (zelená farba) a tretiemu úbežníku **w** (modrá farba) tvoriacimi súradný systém snímanej situácie. . . . . 29
4.13 Jednoduchá schéma komponentu zabezpečujúceho klasifikáciu vozidiel. . . . 30
4.14 Schéma komponentu zabezpečujúceho dopočítanie mierky scény. . . . . 30
4.15 Kľúčové body na vozidle použité pri získavaní mierky. (Obr. prevzatý z [27]) 31
4.16 Vizualizácie zrotovaného, zväčšeného a posunutého modelu na mieste zodpovedajúcom výskytu vozidla. . . . . 32
4.17 Schéma podsystému zabezpečujúceho meranie rýchlosti prechádzajúcich vozidiel. . . . . 32
4.18 Jednoduchá schéma komponentu zabezpečujúceho trackovanie prechádzajúcich vozidiel. . . . . 33
4.19 Jednoduchá schéma komponentu zabezpečujúceho dopočítanie rýchlosti prechádzajúcich vozidiel. . . . . 33
4.20 Ukážka demonštratívneho grafického výstupu zobrazujúceho nameranú rýchlosť prechádzajúcich vozidiel. . . . . 34

6.1 Schéma nahrávania a získavania dát z datasetu *BrnoCompSpeed*. (Obr. prevzatý z [23]) . . . . . 38
6.2 Príklad manuálne nameraných vzdialeností medzi značkami na rovine cesty. (Obr. prevzatý z [23]) . . . . . 39
6.3 Kumulatívny histogram absolútnych chýb jednotlivých verzií implementovaného systému a porovnávaných systémov. . . . . 44

# Kapitola 1

# Úvod

V dnešnej dobe sa každým dňom nachádza na cestných komunikáciách viac a viac vozidiel. Dôsledkom tohto nárastu sú preplnené cesty a následné dopravné zápchy. Ďalším nepriaznivým následkom je negatívny vplyv na životné prostredie. Aby bolo možné tieto vzniknuté problémy s dopravou riešiť, je potrebné dopravu najprv analyzovať.

S analýzou dopravy súvisí aj táto práca. Konkrétne sa táto práca venuje meraniu rýchlosti automobilov z dopravnej kamery. Cieľom tejto práce je poskytnúť systém, ktorý umožní spracovanie videa z dopravnej kamery, jej automatickú kalibráciu a následný výpočet rýchlosti automobilov prechádzajúcich pred touto kamerou. Pod automatickou kalibráciou je myslené, že výsledný systém nebude potrebovať žiadne manuálne zadávané vstupy alebo parametre na svoju funkčnosť. Z výstupu dopravnej kamery (videa) sám určí polohu a orientáciu kamery aj súradný systém zachytávanej scény. Automatická kalibrácia umožní systému spracovávať videá z klasických dopravných kamier bez ohľadu na charakter snímanej cesty/vozovky. Systém teda nepotrebuje dopredu poznať počet jazdných pruhov, prítomnosť, resp. neprítomnosť vodiacich čiar alebo iných značení. Jediným predpokladom je približne rovný tvar cesty. Systém nedokáže automaticky kalibrovať kamery, ktoré snímajú ostré zákruty alebo križovatky.

Čo sa týka merania rýchlosti prechádzajúcich vozidiel, tak navrhovaný systém potrebuje jedine údaj o rýchlosti snímania danej kamery, prípadne časový údaj prislúchajúci ku každej snímke zachytenej kamerou. Detekcia aj výpočet rýchlosti automobilov bude následne prebiehať automaticky.

Navrhnutý a implementovaný systém by mal vo výsledku dosahovať presnosť predošlých riešení a zároveň zjednodušiť a vylepšiť niektoré z metód, ktoré boli doposiaľ použité. Taktiež by mal celý proces kalibrácie a merania zlúčiť do jedného uzatvoreného systému. Od existujúcich systémov sa bude tento systém odlišovať aj v tom, že bude optimalizovaný vzhľadom na čas potrebný k automatickej kalibrácii. Zámerom je teda vytvoriť systém, ktorý nebude musieť na získanie kalibračných parametrov niekoľkokrát spracovať celé video z dopravnej kamery.

Existujúce riešenia, z ktorých bude výsledný systém vychádzať, sú popísané v kapitole 2. Významné algoritmy a metódy detekcie a klasifikácie, ktoré budú použité v riešení, sú obsiahnuté v kapitole 3. Podrobný návrh výsledného systému sa nachádza v kapitole 4 a detaily samotnej implementácie sú obsiahnuté v kapitole 5. Kapitola 6 obsahuje popis testovania a vyhodnotenie navrhnutého a následne implementovaného riešenia a v kapitole 7 je spísaný záver tejto diplomovej práce a plány do budúcna.

# Kapitola 2

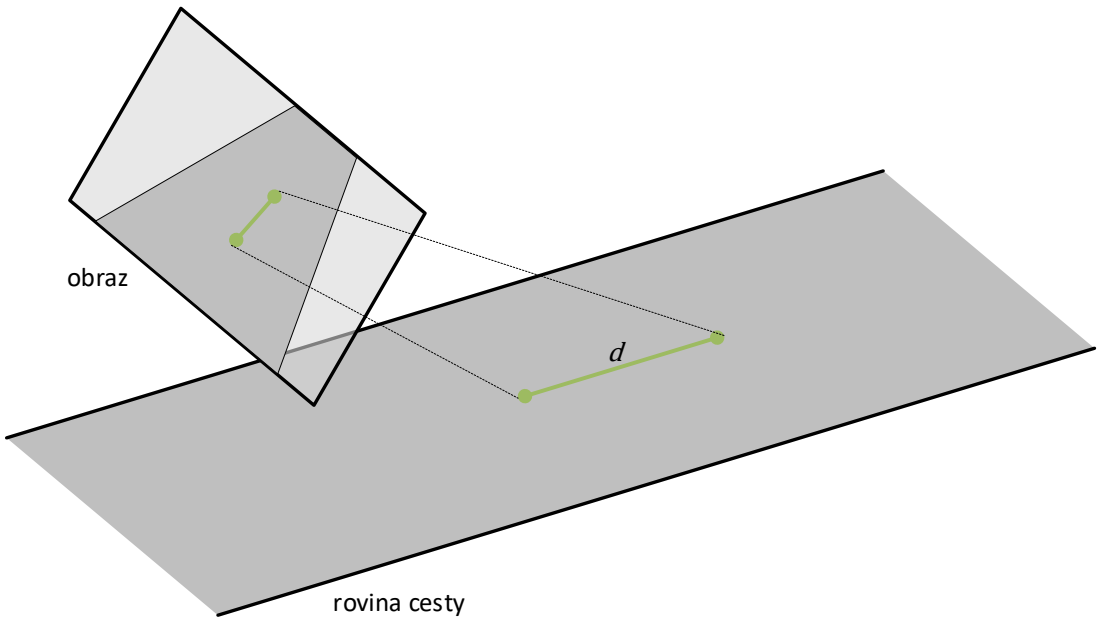
# Existujúce riešenia

V tejto kapitole sú popísané existujúce riešenia, resp. čiastočné riešenia problému, ktorým sa zaoberá táto práca. Opísané budú ich výhody aj nevýhody a najpodrobnejšie budú rozobrané riešenia od Sochora a kol. [22] a Dubskej a kol. [9], na ktorých bude z veľkej časti stavať aj výsledné riešenie tejto práce.

## 2.1 Kalibrácia dopravnej kamery

Pre určenie rýchlosti vozidla je potrebné reflektovať vzdialenosť, o ktorú sa dané vozidlo posunulo v rámci obrazu, na vzdialenosť, ktorú prešlo v reálnej situácii ($d$ na obrázku 2.1). Aby bolo možné hľadanú vzdialenosť určiť pre ľubovoľné dva body v rámci obrazu, je potrebné dopravnú kameru nakalibrovať.

Kalibrácia dopravnej kamery je pre riešenie problému tejto práce veľmi významná. Pre správne meranie rýchlosti prechádzajúcich vozidiel je potrebná vysoká presnosť kalibrácie.




Obr. 2.1: Vzdialenosť dvoch bodov v obraze premietnutá na rovinu cesty.

Nepresnosti, ktoré vzniknú pri kalibrácii kamery, totižto priamo ovplyvňujú presnosť nameranej rýchlosti. Vo všeobecnosti kalibrácia kamery zahŕňa riešenie perspektívneho zobrazenia, rôznych natočení kamery, neznámej vzdialenosti medzi kamerou a základnou rovinou cesty a prípadne aj radiálneho a tangenciálneho skreslenia. Zvyčajne je potrebné získať vnútorné a vonkajšie parametre kamery spolu s mierkou scény alebo vzdialenosťou kamery od roviny cesty/zeme. [23]

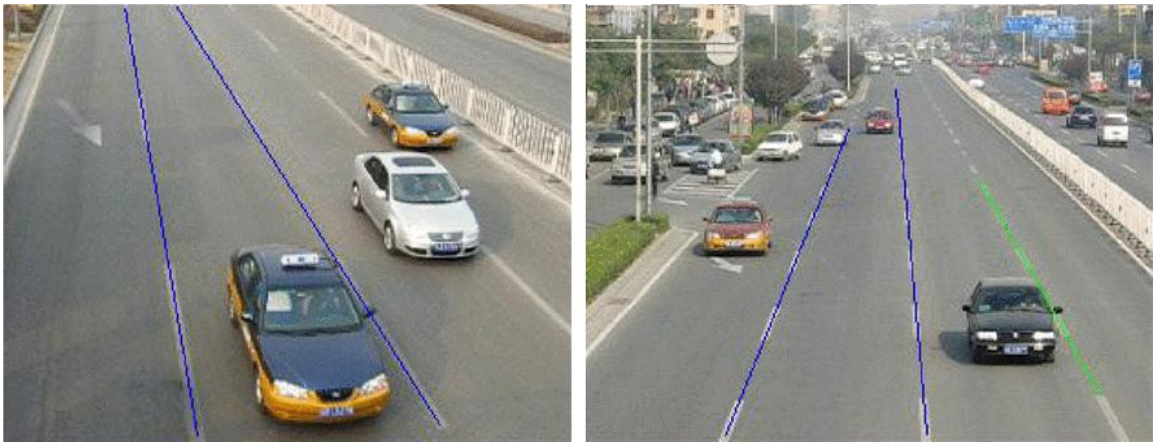
Taktiež je dôležité, aby bola kalibrácia plne automatická, a teda nevyžadovala žiadne vstupy od používateľa. V niektorých existujúcich riešeniach sa vyskytuje len čiastočne automatická kalibrácia dopravnej kamery. Ďalej ju budeme označovať pojmom manuálna kalibrácia. Tá je viac alebo menej závislá na vstupoch používateľa. Tým je výrazne obmedzená rýchlosť a automatickosť spracovávania. Jednak je potrebná prítomnosť zaškoleného personálu pri kalibrovaní výstupu z kamery a okrem toho manuálna kalibrácia často vyžaduje prítomnosť personálu aj v teréne, kde je potrebné namerať a vyznačiť potrebné údaje, ktoré sú následne pri kalibrácii z videa použité. To obmedzuje možnosť využiť na spracovávanie rôzne kamery, keďže každá z nich musí byť samostatne kalibrovaná. Z týchto dôvodov je aj pri tejto práci kladený veľký dôraz na automatickú kalibráciu, ktorá umožní zobrať výstup z väčšej množiny kamier a automaticky ho spracovávať pre potrebné účely.

### 2.1.1 Manuálna kalibrácia

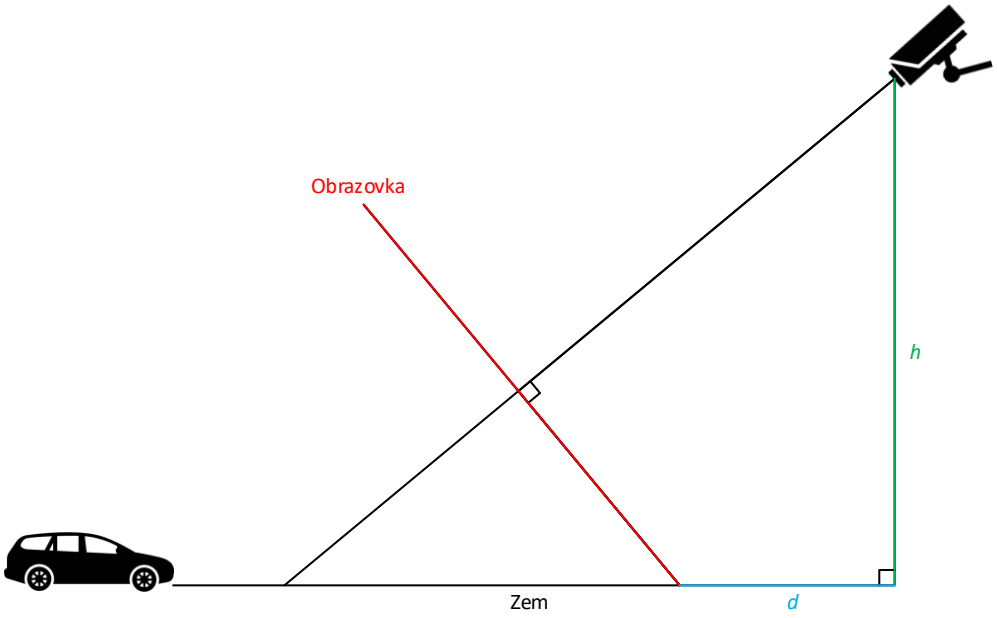
V tejto časti sa zameriame na existujúce riešenia, ktorých kalibrácia kamery neprebieha plne automaticky. Rozoberieme jednotlivé riešenia a ich požiadavky na vstupy od užívateľa a údaje, ktoré je potrebné poznať pred spustením kalibrácie. Manuálne vstupy sú vo väčšine prípadoch potrebné najmä na určenie mierky (koľko pixelov zodpovedá v skutočnosti koľkým metrom).

Manuálna kalibrácia sa vyskytuje napríklad v riešení od Wanga a kol. [26]. Toto riešenie vyžaduje vstup v podobe vyznačenia vodiacich čiar (viď. obrázok 2.2) a znalosť šírky jazdného pruhu. Taktiež je potrebné zadať buď výšku, v ktorej sa kamera nachádza, alebo vzdialenosť medzi vyznačenými značkami na zemi, ktoré sú paralelné k jazdným pruhom. Tieto parametre vyžaduje pre výpočet mierky aj riešenie You a Zheng [28].

Riešenie od Dawsona a Birchfielda [6] taktiež vyžaduje pre správnu funkčnosť kalibrácie vstupné údaje od užívateľa. Nepotrebuje vyznačiť vodiace čiary, tie deteguje automaticky,



Obr. 2.2: Vyznačenie vodiacich čiar použité pre kalibráciu kamery. (Obr. prevzatý z [26])




Obr. 2.3: Nákres požadovaných parametrov pre riešenie od Paia a kol. [19].

ale potrebuje vedieť šírku jazdného pruhu a dĺžku vodiacej čiary. Šírku jazdného pruhu vyžaduje napríklad aj riešenie od Songa a Taia [25].

Manuálnou kalibráciou je obmedzené aj riešenie od Paia a kol. [19], ktoré má medzi požadovanými parametrami výšku, v ktorej sa nachádza kamera ($h$ na obrázku 2.3), a vzdialenosť medzi polohou kamery a príslušnou pozíciou spodného okraja snímky ($d$ na obrázku 2.3).

Čo sa týka riešenia od Daileyho a kol. [5], tak toto riešenie vyžaduje znalosť priemernej dĺžky vozidla. Okrem toho, že je tým obmedzená automatickosť kalibrácie, je tým do značnej miery ovplyvnená aj presnosť. Ako aj autori uvádzajú, vo výsledku sa potom jedná len o hrubý odhad rýchlosti prechádzajúcich vozidiel.

S iným prístupom prišli Schoepflin a Dailey [21]. Rovnako ako v predošlom riešení ale celý systém nepracuje úplne automaticky, vychádza totiž zo zadaných údajov o priemernej rýchlosti prechádzajúcich vozidiel. Druhou možnosťou je zadať namerané rozmery vyznačeného objektu.

Riešenie od Doa a kol. [7] vyžaduje prítomnosť personálu v teréne. Pre kalibráciu kamery je potrebné na vozovku vyznačiť a zamerať rovnostranný trojuholník (viď. obrázok 2.4). Na základe znalosti reálnych rozmerov vyznačeného rovnostranného trojuholníka je následne možné dopočítať vonkajšie a vnútorné parametre kamery. Z údajov nameraných v reálnom prostredí, ktoré sú následne využité pri kalibrácii, taktiež vychádzajú aj riešenia od Luvizona a kol. [17, 18].

Chaperon a kol. [4] taktiež predstavili riešenie súvisiace s kalibráciou kamery. Toto riešenie síce nesúvisí s analýzou dopravy, ale zaoberá sa, rovnako ako táto práca, problémom získania vnútorných a vonkajších parametrov kamery. Kalibrácia avšak, tak isto ako v riešeniach uvedených vyššie, nie je plne automatická. Pracuje na základe vyhľadávania zhôd medzi užívateľom zakreslenými bodmi a čiarami.


Obr. 2.4: Scéna s rovnostranným trojuholníkom vyznačeným na zemi. (Obr. prevzatý z [7])

### 2.1.2 Automatická kalibrácia

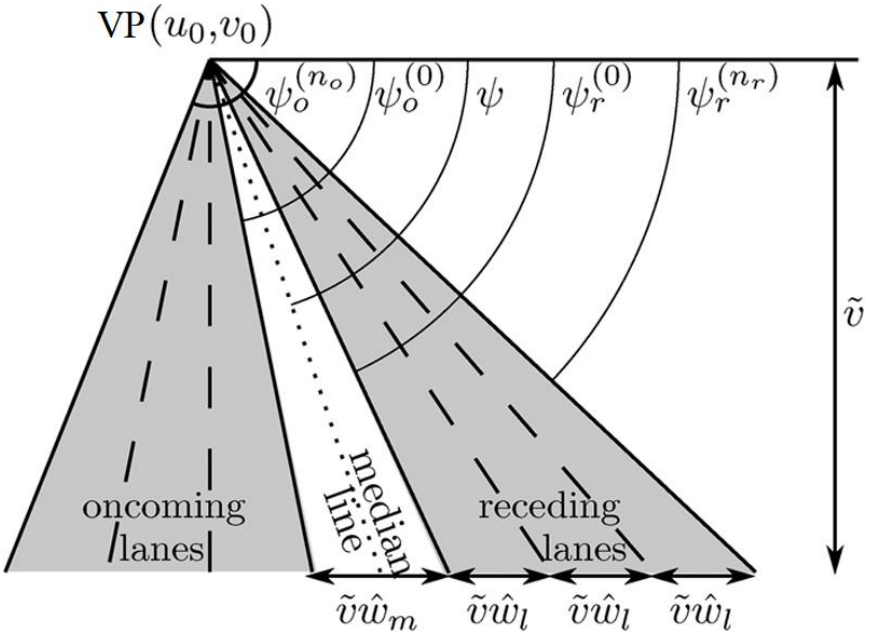
Plne automatická kalibrácia dopravnej kamery sa nachádza v prácach Dubskej a kol. [9] a Sochora a kol. [22]. Riešenia z týchto prác nepotrebujú žiadne merania v reálnom svete. Taktiež nepožadujú žiadne dodatočné vstupy ani informácie od užívateľa. Nepredpokladajú žiadne apriórne znalosti o type prechádzajúcich vozidiel ani o charaktere snímanej cesty (počet a šírka jazdných pruhov, viditeľnosť vodiacich čiar alebo iného značenia či ďalších špecifických vlastností). Tieto riešenia sú zostavené pre bežné scenáre v monitorovaní dopravy. Automaticky určia parametre kompenzácie radiálneho skreslenia a vyriešia kalibráciu vnútorných a vonkajších parametrov kamery (orientáciu a pozíciu kamery). Jediným obmedzením týchto riešení je, že predpokladajú približne rovný tvar cesty.

Rozdiel medzi týmito dvomi metódami je v tom, akým spôsobom je získaná mierka. Zatiaľ čo pri riešení od Dubskej a kol. [9] je mierka získavaná pomocou trojdimenzionálnych ohraničujúcich boxov a stredných hodnôt rozmerov vozidiel, v riešení od Sochora a kol. [22] je mierka získavaná pomocou porovnávania vyrenderovaného modelu konkrétneho vozidla so zodpovedajúcim vozidlom detegovaným vo videu.

### 2.1.3 Princíp kalibrácie dopravnej kamery

Väčšina metód kvôli nedostatku informácii, ktoré je možné získať z cestných dát, predpokladá, že hlavný bod sa nachádza v strede obrazu (napr. [22, 9, 21, 25]). Ďalším predpokladom, ktorý sa často vyskytuje v existujúcich riešeniach, je vodorovná čiara horizontu, čiže vrchný okraj snímky kamery (napr. [21, 14, 11]). Tento predpoklad sa ale javí ako veľmi limitujúci, pretože je ťažké nájsť dopravnú kameru, ktorá by ho presne spĺňala. [9]

Takmer všetky riešenia spája detekcia úbežníka (*Vanishing Point* resp. *VP*) zodpovedajúceho smeru pohybu vozidiel. Jednotlivé riešenia sa ale líšia v spôsobe, ako tento úbežník získavajú. Prvou skupinou sú riešenia, ktoré na detekciu úbežníka využívajú jazdné pruhy alebo vodiace čiary (napr. [26, 6, 25, 14, 11]). Automaticky alebo manuálne získané vodiace




Obr. 2.5: Ilustrácia vodiacich čiar a jazdných pruhov zbiehajúcich sa v úbežníku ($VP$). (Obr. prevzatý z [6])

čiary a jazdné pruhy sa zbiehajú v hľadanom úbežníku (viď obrázok 2.5). Nevýhodou týchto riešení je potreba presného a dobre viditeľného značenia vodiacich čiar a taktiež vysoký počet jazdných pruhov.

Druhá skupina nevyužíva pri riešení vodiace čiary vyznačujúce jazdné pruhy na ceste, nakoľko môže značenie chýbať alebo nemusí byť dostatočne výrazné, a tým pádom je nedetegovateľné. Namiesto toho sleduje pohyb vozidiel. Tým sa ale obmedzuje len na kamery s rovnobežnou a paralelnou trajektóriou v dominantnej časti pohľadu. V riešení, ktoré vytvorili Schoepflin a Dailey [21], sa na základe detegovaných vozidiel vytvára mapa aktivity. Z tejto mapy sú následne získané hranice jednotlivých jazdných pruhov (viď. obrázok 2.6). Úbežník je potom, rovnako ako pri prvej skupine riešení, lokalizovaný v bode, kde sa tieto hranice zbiehajú, resp. pretínajú. Nevýhodou tohto riešenia je, že taktiež vyžaduje pre presnosť vysoký počet jazdných pruhov.

Do druhej skupiny patria aj riešenia od Dubskej a kol. [9] a Sochora a kol. [22], ktoré využívajú pre získanie úbežníku sledovanie kľúčových bodov na prechádzajúcich vozidlách. Čiary zostavené z pohybu týchto kľúčových bodov sú následne naakumulované v *diamond space* akumulátore [8], ktorý nájde bod, v ktorom sa nachádza hľadaný úbežník. Podrobnejšie vysvetlenie *diamond space* akumulátoru sa nachádza v podkapitole 3.1.1.

Niektoré riešenia pracujú len s jedným úbežníkom, sú navrhnuté tak, že pre kalibráciu ďalší úbežník nepotrebujú (Dawson a Birchfield [6]), alebo pracujú s predpokladom, že je kamera nahnutá len v osi $x$, a tým pádom druhý úbežník leží v nekonečne a nemusia ho získavať (Grammatikopoulos a kol. [12]). Iné riešenia (Dubská a kol. [10]) pracujú aj s druhým úbežníkom, prípadne aj s tretím (napr. Sochor a kol. [22], Dubská a kol. [9]). Tieto riešenia získavajú druhý úbežník pomocou detegovania hrán na vozidlách, ktoré sú odklonené od prvého úbežníku. Na odvodenie úbežníku z detegovaných hrán je následne využitý *diamond space* akumulátor. V tomto prípade je hľadaný úbežník kolmý na smer pohybu

Obr. 2.6: Mapa aktivity a získané hranice jazdných pruhov (nad snímkami z rozdielnych kamier). (Obr. prevzatý z [21])

vozidiel a paralelný s rovinou zeme. Posledný úbežník je možné následne, za predpokladu, že sa hlavný bod nachádza v strede obrazu, dopočítať.

Iný prístup sa nachádza v riešení od Youa a Zhenga [28]. Toto riešenie tiež pracuje s dvoma úbežníkmi, ale po získaní prvého úbežníka, ktorý zodpovedá smeru pohybu vozidiel, sa hľadá úbežník kolmý na rovinu zeme. Na jeho získanie je využitá detekcia stĺpov a chodcov.

S pomocou získaných úbežníkov je možné následne definovať súradný systém obrazu zachyteného kamerou a aj vonkajšie a vnútorné parametre kamery. Akonáhle je kamera nakalibrovaná a je k dispozícii mierka scény, je možné určiť rovinu cesty. Následne je možné jednoduchým spôsobom použiť rôzne aplikácie, ako napríklad počítanie vozidiel a ich klasifikácia, detekcia dopravných zápch, meranie rýchlosti atď.

## 2.2 Meranie rýchlosti prechádzajúcich vozidiel

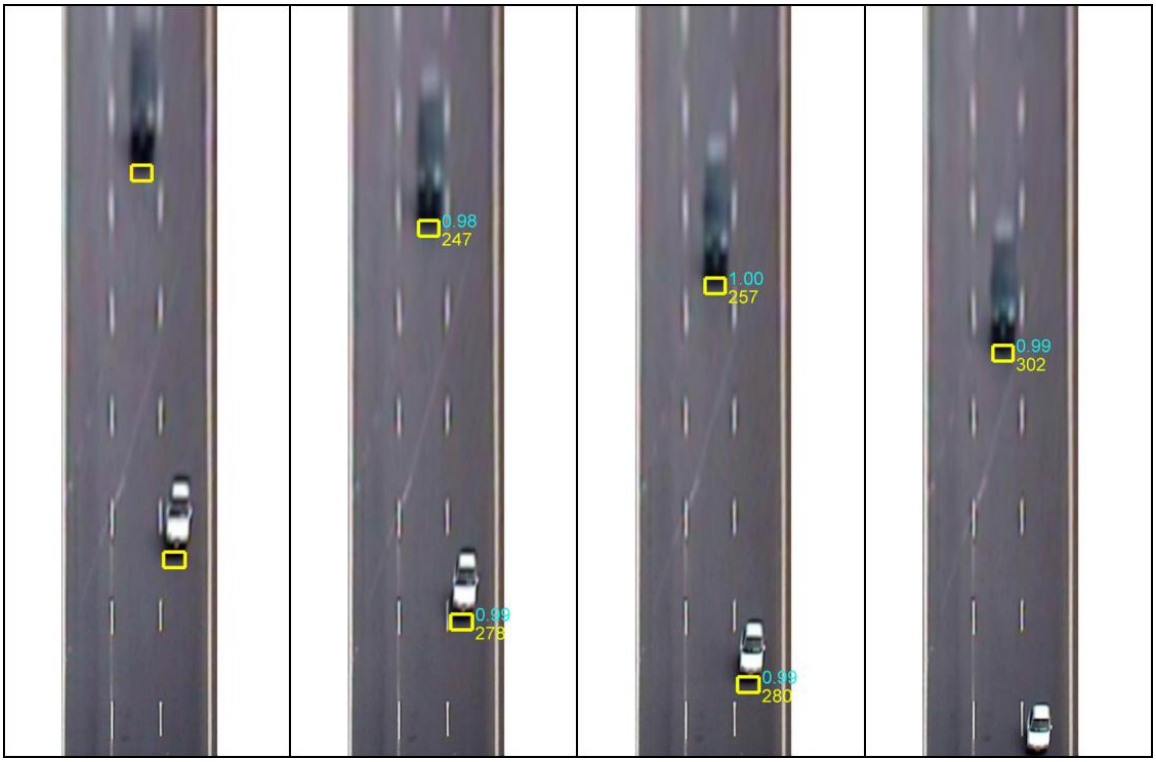
Čo sa týka princípu merania rýchlosti prechádzajúcich vozidiel, tak ten je vo väčšine riešení zhodný. V zásade je rýchlosť dopočítaná na základe toho, o akú vzdialenosť sa vozidlo pohlo medzi jednotlivými snímkami. Jednotlivé riešenia sa líšia v tom, z akých bodov vypočítavajú prejdenú vzdialenosť a po akom počte snímok je rýchlosť prepočítavaná.

Napríklad v riešení od Sochora a kol. [22] je rýchlosť dopočítavaná na základe referenčných bodov na vozidle a časovej značky pre každý z týchto bodov. Tieto referenčné body sú projektované na rovinu zeme a rýchlosť je následne dopočítaná ako stredná hodnota rýchlostí medzi po sebe idúcimi pozíciami v čase. Pre vyššiu stabilitu nie sú použité snímky, ktoré idú priamo za sebou, ale s rozostupom štyroch snímok.

Luvizon a kol. [17] používajú na výpočet rýchlosti detekciu poznávacích značiek. Rýchlosť vypočítavajú z vektora pohybu, ktorý zaznamenáva pohyb sledovaných bodov na poznávacej značke. Vzdialenosť, ktorú vozidlo prešlo, predstavuje priemer vzdialeností, o ktoré sa posunuli jednotlivé sledované body. Rýchlosť vozidiel je vypočítavaná z každej dvojice susediacich snímok.

Grammatikopoulos a kol. [12] využívajú na určenie prejdenej vzdialenosti sledovanie najnižšej časti vozidla (koniec predného nárazníka, viď obrázok 2.7), keďže sa nachádza najbližšie k rovine zeme. He a Yung [14] používajú vzdialenosť medzi spodnými okrajmi

detegovaných masiek vozidiel. Rovnaký prístup je použitý aj v riešení od Schoepflina a Daileyho [21].




Obr. 2.7: Sledovanie zistených profilov (snímka úplne vľavo) v troch po sebe nasledujúcich snímkach. (Obr. prevzatý z [12])

# Kapitola 3

# Použité metódy

V tejto kapitole sú popísané metódy a teoretické informácie, ktoré sú podstatné pre riešenie problému, ktorým sa táto práca zaoberá. Dôležitou súčasťou kalibrácie dohľadovej kamery je presná detekcia úbežníkov. Pri nej je možné využiť *diamond space* akumulátor. Princíp fungovania tejto metódy je popísaný v podkapitole 3.1. Ďalšou dôležitou súčasťou pri meraní rýchlosti prechádzajúcich vozidiel je detekcia samotných vozidiel. Na detekciu vozidiel existuje mnoho metód a algoritmov. V tejto práci sme sa rozhodli použiť moderné konvolučné detektory objektov. Z tohto dôvodu obsahuje podkapitola 3.2 stručné uvedenie do problematiky konvolučných neurónových sietí. V podkapitole 3.3 je popísaný princíp fungovania samotných konvolučných detektorov objektov so zameraním na *SSD* (*Single Shot Detector*) a *Faster R-CNN* (*Regions with Convolutional Neural Network*). Konvolučným neurónovým sieťam sa venuje aj podkapitola 3.4. Konkrétne ide o konvolučné neurónové siete s hlbokým *residuálnym* učením. Popisuje ich využitie pri klasifikácii obrazu. Tá bude potrebná pre klasifikovanie jednotlivých typov vozidiel, na ktoré sa bude viazať určovanie mierky scény.

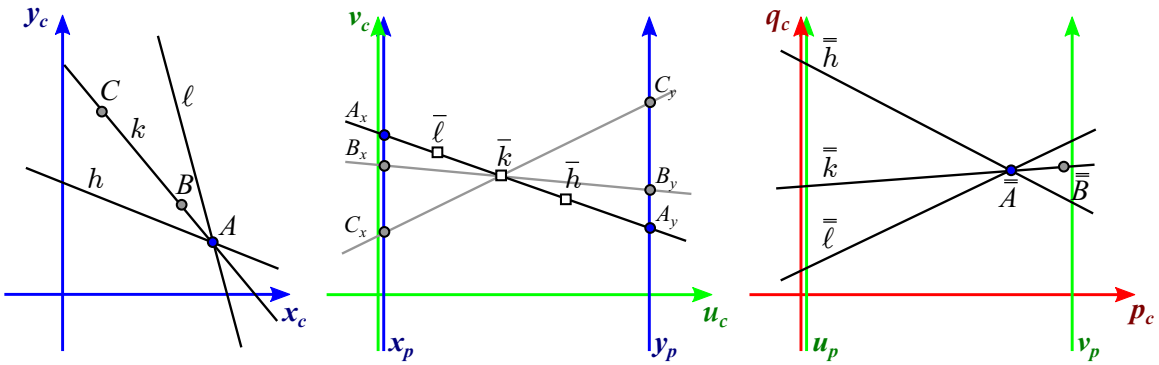
## 3.1 Detekcia úbežníkov

Ako už bolo spomenuté vyššie, presná detekcia úbežníkov je dôležitá pre správne fungovanie riešenia problému, ktorým sa táto práca zaoberá. Úbežníky je možné detegovať viacerými spôsobmi (viď. 2.1.3). V tejto práci sa budeme zaoberať hlavne detekciou úbežníkov s využitím *diamond space* akumulátoru, a to kvôli robustnosti tejto metódy a tomu, že nepredpokladá žiadne prekvizitné znalosti pre správnu kalibráciu dohľadovej kamery.

### 3.1.1 Diamond Space akumulátor

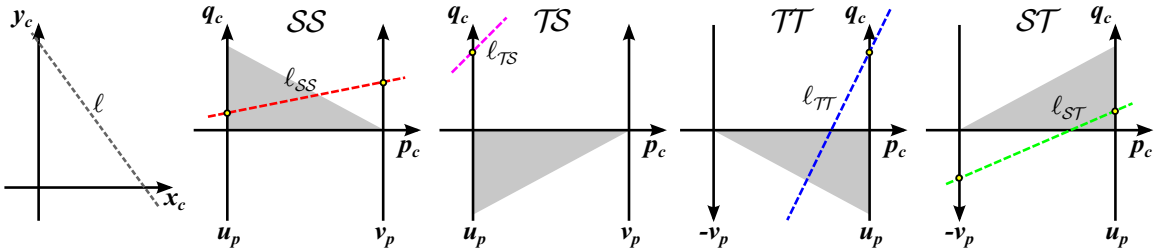
Pri detekcii úbežníkov je možné využiť *diamond space* akumulačnú schému založenú na Houghovej transformácii. Tento algoritmus bol navrhnutý Dubskou a Heroutom [8] a následne úspešne použitý v riešeniach od Dubskej a kol. [9] a Sochora a kol. [22].

Princíp tejto metódy je v mapovaní celej 2D projekčnej roviny do konečného priestoru, ktorý je označovaný ako *diamond space*. Toto mapovanie prebieha pomocou lineárneho mapovania priamok po častiach za použitia paralelných súradníc (súradný systém, kde sú osi navzájom paralelné). 2D bod $[x, y]$ je potom v tomto súradnom systéme reprezentovaný priamkou pretínajúcou súradnicové osi v hodnotách $x$ a $y$. Mapovanie prebieha takýmto transformovaním každého bodu na priamku a následne znova na bod (podobne pri transformácii priamka-bod-priamka). Príklad postupnej transformácie je zobrazený na obrázku 3.1.




Obr. 3.1: Dve kaskádové transformácie bodov a priamok do paralelných súradníc. (Obr. prevzatý z [9])

Transformácia do systému paralelných súradníc môže byť vykonaná dvomi rôznymi spôsobmi. Osi v systéme paralelných súradníc môžu byť buď orientované rovnakým, alebo opačným smerom. Prípad, keď sú osi orientované rovnakým smerom (obrázok 3.1), označujeme $\mathcal{S}$ (*straight*), a prípad s osami orientovanými v opačnom smere označujeme $\mathcal{T}$ (*twisted*). Kompozitné mapovanie môže byť teda vykonávané štyrmi rôznymi spôsobmi: $\mathcal{S} \circ \mathcal{S}$, $\mathcal{T} \circ \mathcal{S}$, $\mathcal{T} \circ \mathcal{T}$ a $\mathcal{S} \circ \mathcal{T}$. Transformáciu priamky týmito štyrmi spôsobmi možno vidieť na obrázku 3.2.

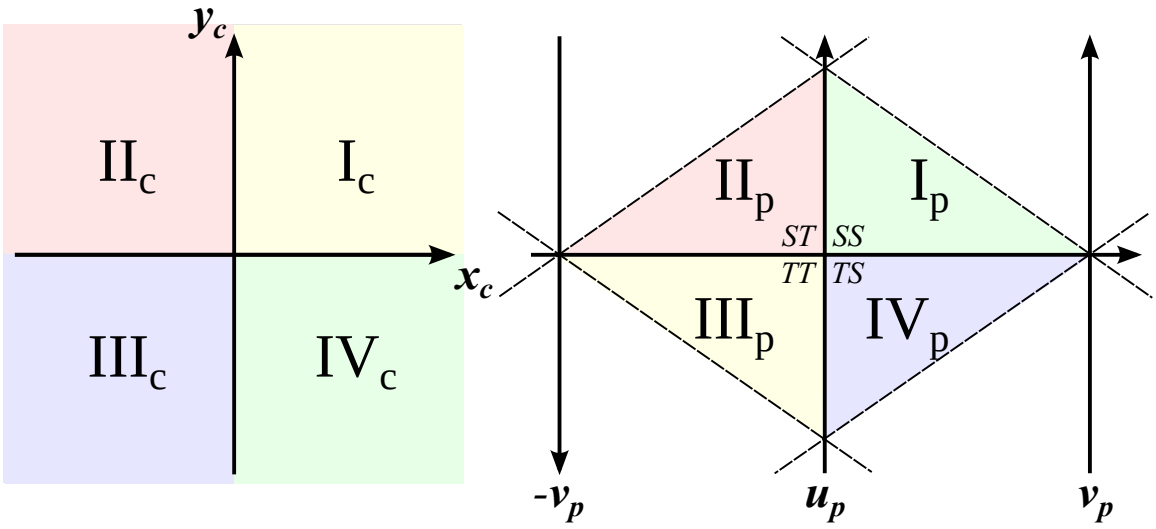



Obr. 3.2: Reprezentácia priamky v rozdielnych transformáciách. Iba tmavé trojuholníkové podpriestory sú súčasťou výsledného *diamond space*. (Obr. prevzatý z [9])

Každé z vyššie uvedených mapovaní predstavuje transformáciu jedného nekonečného priestoru do druhého. Avšak aplikovanie každého z týchto mapovaní na jeden z kvadrantov pôvodnej projekčnej roviny vedie ku konečnej doméne transformácie. Trojuholníkové podpriestory z obrázku 3.2 je možné pripojiť k sebe, čím vzniká *diamond space* (viď. obrázok 3.3).

Priamka s homogénnymi súradnicami $(a, b, c)$ z 2D projekčnej roviny je v *diamond space* reprezentovaná zloženou úsečkou. Pričom počet kvadrantov, ktoré pôvodná priamka pretína, zodpovedá počtu segmentov, ktoré bude mať krivka, ktorá ju reprezentuje. V prípade, že pôvodná priamka pretína iba dva kvadranty (vodorovné priamky, zvislé priamky, priamky pretínajúce počiatok súradného systému), degeneruje vždy jeden segment na bod.

Takáto transformácia mapuje body v nekonečne aj bežné body na bežné body a je použitá ako parametrizácia v Houghovej transformačnej schéme. Priamky v doméne obrazu sú transformované do *diamond space* a výsledná krivka je akumulovaná do transformačnej schémy – každý pixel krivky je inkrementovaný. Následne je v *diamond space* nájdené globálne maximum, ktoré zodpovedá hľadanému úbežníku – miesto, ktorým prechádza najviac priamok. Bod z *diamond space* s homogénnymi súradnicami $[p, q, 1]$ je následne spätne projektovaný do pôvodnej projekčnej roviny. [9]




Obr. 3.3: Kvadranty paralelných priestorov (vpravo) zodpovedajúce kvadrantom pôvodného nekonečného Karteziánskeho priestoru (vľavo). (Obr. prevzatý z [9])

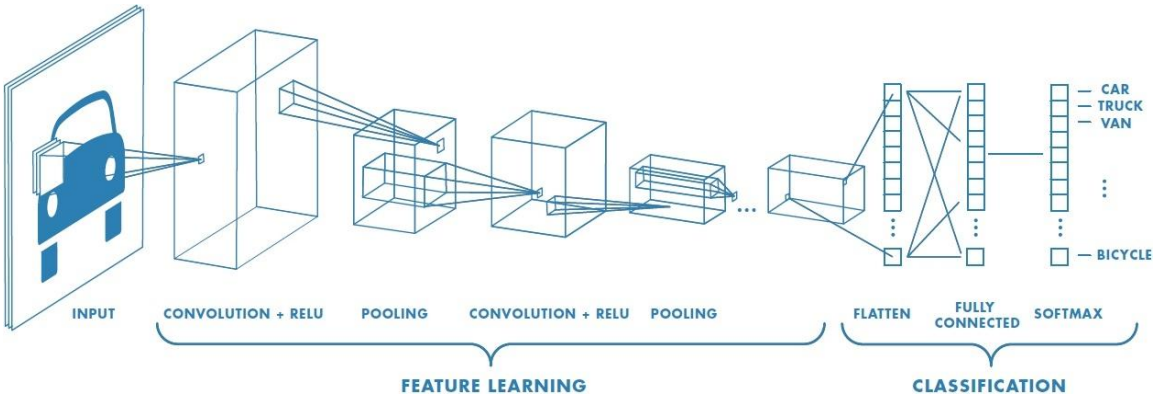
## 3.2 Konvolučné neurónové siete

Konvolučné neurónové siete (*Convolutional Neural Networks – CNN*) sú špeciálnym druhom viacvrstvových neurónových sietí, ktoré boli navrhnuté pre rozpoznávanie dvojrozmerných obrazových vzorov priamo z pixelov obrázkov s minimálnym predspracovaním. *CNN* sa často používajú na rozpoznávanie objektov a scén. Okrem toho sa používajú aj na detekciu a segmentáciu objektov. Minimálne predspracovanie predstavuje to, že sa *CNN* učia priamo z obrazových dát, čím eliminujú potrebu manuálnej extrakcie čŕt. [1]

Konvolučná neurónová sieť môže mať desiatky alebo stovky vrstiev. Každá z týchto vrstiev sa pritom učí detegovať rôzne črty v obraze. Na trénovacie obrázky sú aplikované filtre pri rôznych rozlíšeniach a výstup každého konvolovaného obrázku predstavuje vstup pre ďalšiu vrstvu. Filtre môžu začínať veľmi jednoduchými črtami, ako sú jas a hrany, a naberať na komplexnosti k črtám, ktoré pri postupovaní vrstvami jednoznačne definujú objekt. Príklad takejto konvolučnej neurónovej siete sa nachádza na obrázku 3.4.

Jednou z metód na vytvorenie konvolučnej neurónovej siete slúžiacej na rozpoznávanie objektov je jej naučenie od úplných základov. Architekt takejto siete musí definovať počet vrstiev, váhy učenia a počet filtrov spolu s inými nastaviteľnými parametrami. Trénovanie presného modelu od úplných základov taktiež vyžaduje obrovské množstvo dát, rádovo v miliónoch vzoriek, z čoho vyplýva aj obrovské množstvo času, ktoré môže byť potrebné na natrénovanie takéhoto modelu.

Alternatívu predstavuje použitie predtrénovaného modelu konvolučnej neurónovej siete k automatickej extrakcii čŕt z novej dátovej sady. Táto metóda, nazývaná *transfer learning*, predstavuje vhodný spôsob, ako aplikovať hlboké učenie bez obrovskej množiny dát a dlhého času potrebného na výpočet a trénovanie. Z týchto dôvodov bude aj v tejto práci uprednostnené využitie predtrénovaných modelov pred vytváraním nových modelov od úplných základov.



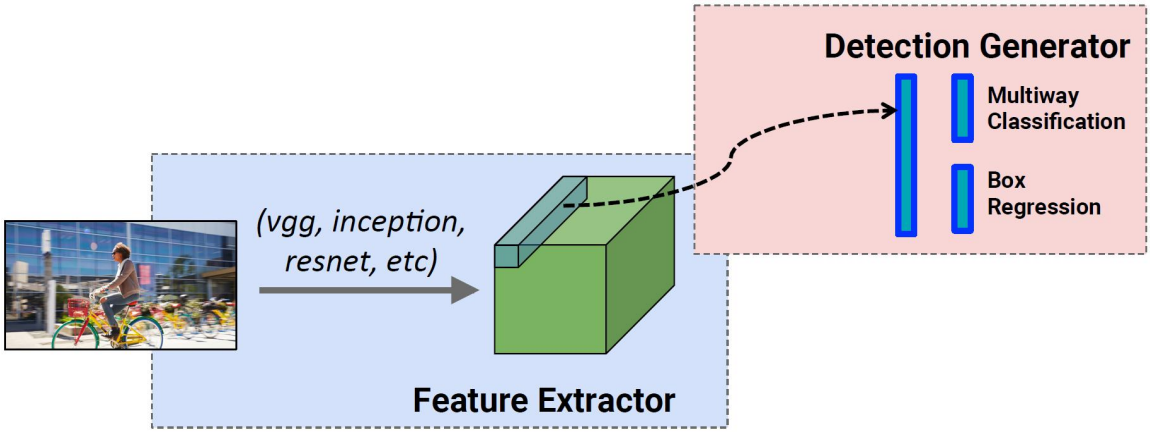

Obr. 3.4: Príklad siete s mnohými konvolučnými vrstvami. (Obr. prevzatý z [1])

## 3.3 Detekcia vozidiel

V detekcii objektov nastal v posledných rokoch veľký posun. Jedným z hlavných dôvodov je využitie konvolučných neurónových sietí (*CNN*). Detektory používajúce *CNN* jednoznačne vedú vo vysoko kvalitnej detekcii objektov. Presnosť týchto detektorov ale nie je jediným smerodajným faktorom. Vysoký význam má pre nás aj rýchlosť a pamäťová náročnosť, s akou sú schopné pracovať. V tejto práci sme sa zamerali hlavne na *SSD* (*Single Shot Detector*)[16] a *Faster R-CNN* (*Regions with Convolutional Neural Network*)[20]. Pri ich porovnaní sme vychádzali z výsledkov práce Huanga a kol. [15]

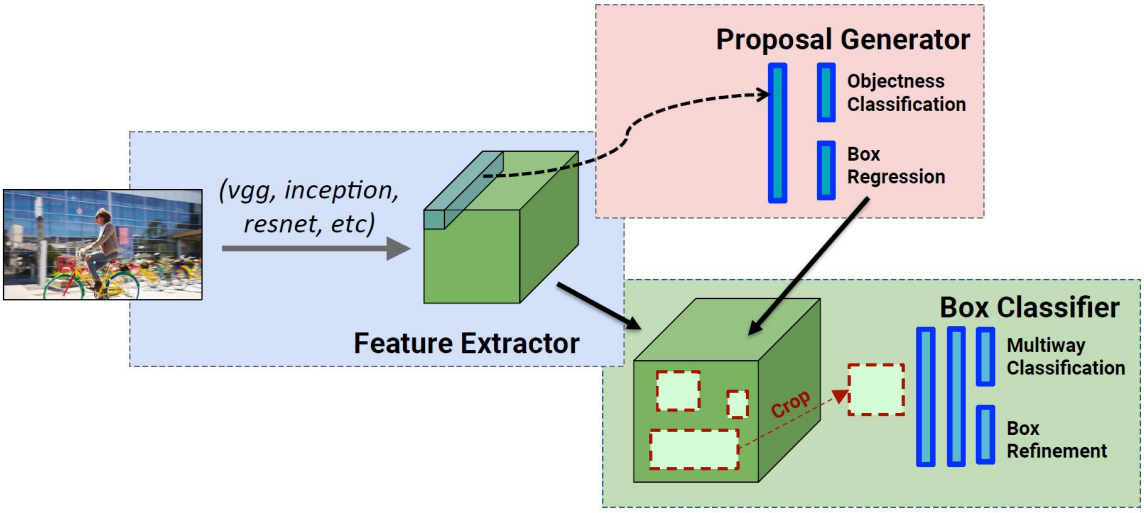
### 3.3.1 SSD

Architektúra *SSD* využíva jednu doprednú konvolučnú sieť k priamej predikcii tried (v prípade tejto práce je detegovaná len jedna trieda – vozidlo) a ofsetov ukotvenia (viď. obrázok 3.5). Táto architektúra, na rozdiel od *Faster R-CNN*, nepotrebuje druhú fázu klasifikácie tried z návrhového generátora (viď. obrázok 3.6). Toto má za následok vyššiu rýchlosť, ale aj nižšiu presnosť detektora s architektúrou *SSD* ako detektora s architektúrou *Faster R-CNN*. Detailnejšie porovnanie sa nachádza v podkapitole 3.3.3.




Obr. 3.5: Diagram meta-architektúry detektora *SSD* (*Single Shot Detector*). (Obr. prevzatý z [15])

### 3.3.2 Faster R-CNN




Obr. 3.6: Diagram meta-architektúry detektora *Faster R-CNN* (*Regions with Convolutional Neural Network*). (Obr. prevzatý z [15])

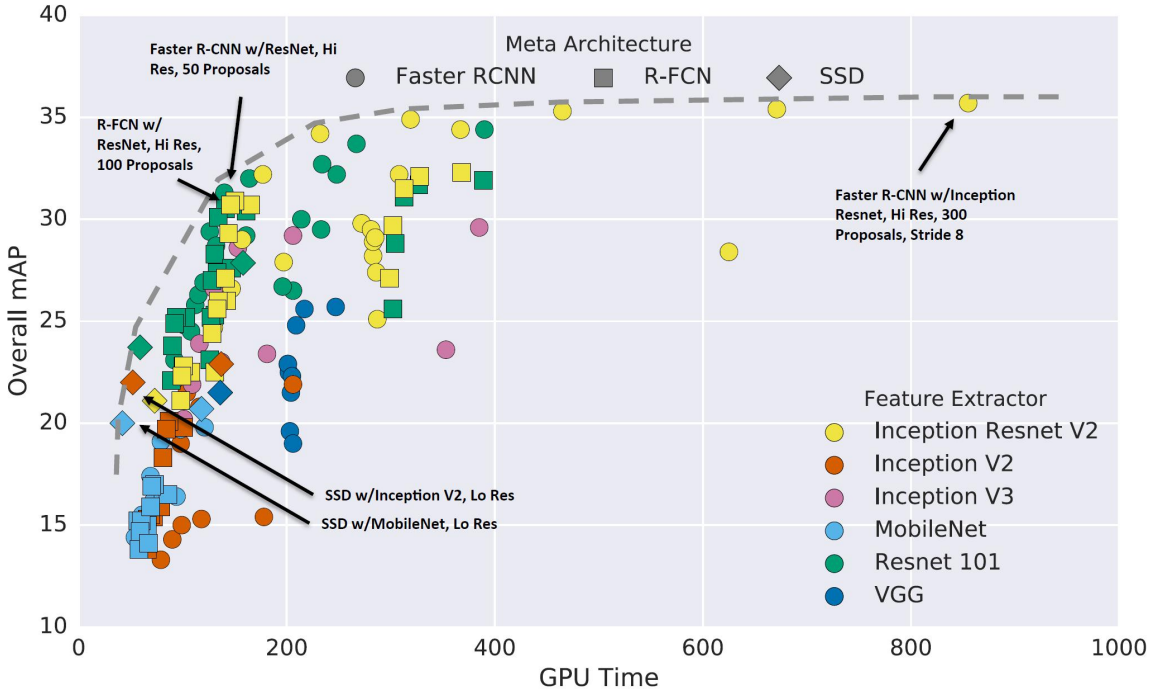
Pri architektúre *Faster R-CNN* prebieha detekcia v dvoch fázach (viď. obrázok 3.6). Prvá fáza sa označuje sieť návrhov oblastí (*Region Proposal Network – RPN*). V tejto fáze sú snímky spracované extraktorom dôležitých čŕt. Získané črty vybranej úrovne sú následne použité na predpovedanie návrhov priradení tried k jednotlivým boxom (*Proposal Generator* na obrázku 3.6). Tie sú predané druhej fáze (*Box Classifier* na obrázku 3.6). Druhá fáza zdokonaľuje predpoveď tried a triednych boxov tým, že využije obdržané návrhy k orezaniu čŕt z mapy čŕt rovnakej úrovne. Orezané črty sa následne zavádzajú do zvyšnej časti extraktora čŕt. Návrhy nemôžu byť orezané priamo zo snímky a následne znova spracované extraktorom čŕt, nakoľko by to viedlo k duplicitným výpočtom. Preto je predpovedanie návrhov spúšťané zvlášť nad každou oblasťou a rýchlosť celého výpočtu priamo závisí od toho, koľko oblastí je navrhnutých v *RPN*.

### 3.3.3 Porovnanie SSD a Faster R-CNN

Huang a kol. [15] vo svojej štúdii porovnali architektúry typu *SSD*, *Faster R-CNN* a *R-FCN* (*Region-based Fully Convolutional Networks*). Okrem porovnania samotných architektúr testovali a porovnávali aj rôzne metódy použité pre implementáciu extraktora čŕt. Výsledky tohto porovnania možno vidieť v grafe na obrázku 3.7. Tento graf porovnáva čas, potrebný pre výpočet, s presnosťou. Jednotlivé architektúry sú rozlíšené tvarom značky a verzie extraktora čŕt sú odlíšené farbami. Pre každý pár (architektúra – extraktor čŕt) je v grafe viacero značiek, keďže porovnávanie prebiehalo aj na rôznych veľkostiach vstupných dát, kroku atď.

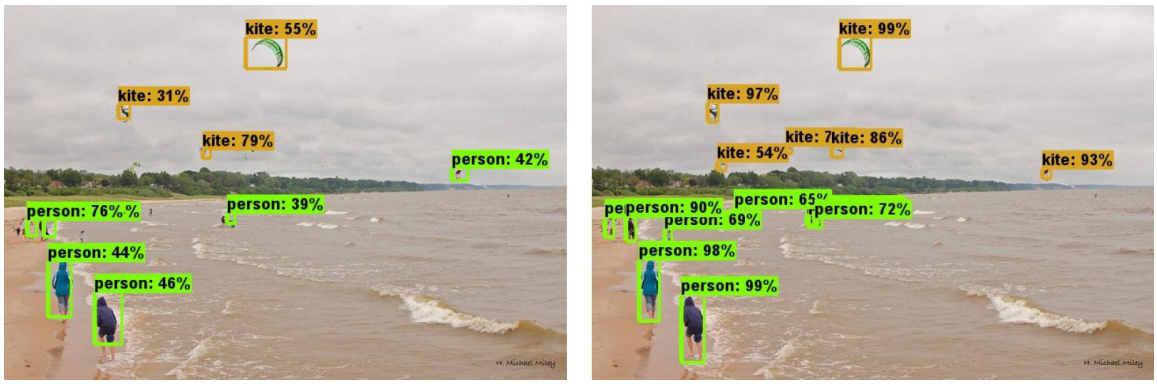
Ako možno v grafe vidieť, najrýchlejšie výsledky dosahovala architektúra *SSD* a najkvalitnejší výstup architektúra *Faster R-CNN*. Podstatnú rolu zohral ale aj extraktor čŕt. Pre ďalšie porovnania, podrobnejšie vyhodnotenie a detailnejší popis jednotlivých extraktorov viď. [15].

Rozdiely vo výstupe medzi najrýchlejšou verziou architektúry *SSD* a verziou architektúry *Faster R-CNN* s najvyššou kvalitou je možné vidieť na obrázku 3.8. Z detegovaných




Obr. 3.7: Porovnanie presnosti jednotlivých architektúr a spôsobu implementácie extraktora čŕt voči času potrebnému na výpočet. (Obr. prevzatý z [15])

objektov na obrázkoch je zrejmé, že výstup sa nelíši len počtom detegovaných objektov, ale aj ich lokalizáciou a ohraničením.




Obr. 3.8: Príklad detekcie osôb (*person*) a šarkanov (*kite*) najrýchlejšiou verziou *SSD* (vľavo) a najpresnejšou verziou *Faster R-CNN* (vpravo). (Obr. prevzatý z [15])

## 3.4 Klasifikácia vozidiel

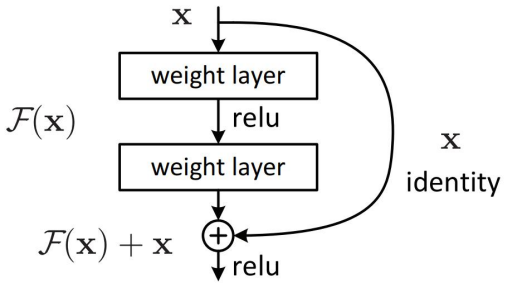
V oblasti klasifikácie obrazu viedli hlboké konvolučné neurónové siete (*CNN*) k sérii prelomov. Prirodzene integrujú nízko-, stredno- a vysokoúrovňové príznaky a klasifikátory do uzatvorenej viacúrovňovej podoby, kde úrovne príznakov môžu byť obohatené počtom naskladaných vrstiev (hĺbkou). Ako je spomínané v práci od Hea a kol. [13], hĺbka siete má

rozhodujúci význam, avšak iba jednoduché pridanie ďalších vrstiev nemusí viesť k lepším výsledkom. So zvyšovaním hĺbky sietí prichádza problém degradácie – presnosť sa nasýti a následne rapídne degraduje. V nasledujúcej podkapitole je popísané riešenie tohto problému.

### 3.4.1 Hlboké residuálne učenie

Problému degradácie sa venuje už spomínaná práca od Hea a kol. [13] a ako riešenie navrhuje *framework* s hlbokým *residuálnym* učením. Namiesto dúfania, že každých pár naskladaných vrstiev priamo zapasuje do požadovaného podkladového mapovania, explicitne nechávajú tieto vrstvy napasovať do *residuálneho* mapovania. Požadované podkladové mapovanie je formálne označené ako $\mathcal{H}(x)$, naskladané nelineárne vrstvy sú napasované do iného mapovania $\mathcal{F}(x) \coloneqq \mathcal{H}(x) - x$. Originálne mapovanie je teda preformulované na $\mathcal{F}(x) + x$. Toto riešenie pracuje s hypotézou, že je jednoduchšie optimalizovať *residuálne* mapovanie ako optimalizovať pôvodné nezaradené mapovanie.

Formuláciu $\mathcal{F}(x)+x$ možno realizovať pomocou dopredných neurónových sietí so skrátenými spojeniami. Skrátené spojenia sú tie, ktoré preskakujú jednu alebo viac vrstiev (*layer*). V tomto riešení skrátené spojenia jednoducho vykonávajú mapovanie identity a ich výstup je pridaný k výstupu z naskladaných vrstiev (viď. obrázok 3.9). Výhodou takýchto spojení je, že nepridávajú na výpočetnej komplexnosti.




Obr. 3.9: Stavebný blok *residuálneho* učenia. (Obr. prevzatý z [13])

Reformulácia mapovania na $\mathcal{F}(x) + x$ je motivovaná problémom degradácie. Ak by totižto pridané vrstvy boli konštruované ako mapovanie identity, tak by hlbšie modely nemali mať vyššiu chybu trénovania ako ich plytšie náprotivky. Problém degradácie naznačuje, že riešitelia môžu mať problémy s aproximáciou mapovania identity pomocou viacerých nelineárnych vrstiev. V prípade, že sú mapovania identity optimálne, je možné, pri preformulovaní na *residuálne* učenie, jednoducho znížiť váhy viacerých nelineárnych vrstiev k nule, a tým sa priblížiť k mapovaniu identity.

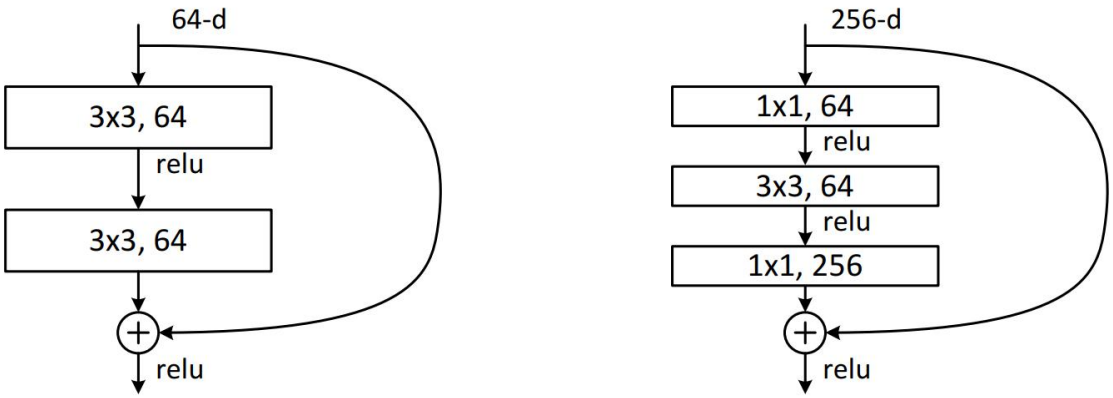
V reálnych prípadoch je nepravdepodobné, že je mapovanie identity optimálne, ale táto reformulácia môže pomôcť stabilizovať problém. Pokiaľ je optimálna funkcia bližšie k mapovaniu identity ako k nulovému mapovaniu, malo by byť jednoduchšie nájsť odchýlky vzhľadom k mapovaniu identity ako naučenie sa funkcie ako úplne novej. [13]

### 3.4.2 Residuálne siete

*Residuálna* sieť *ResNet* (*Residual Network*) predstavuje sieť postavenú na hlbokom *residuálnom* učení. Ku každej dvojici vrstiev konvolučných $3 \times 3$ filtrov je pridané skrátené spojenie

(viď. obrázok 3.10 vľavo). Vďaka tomu nedochádza k degradácii navyšovaním počtu vrstiev ako pri klasickej sieti bez skráteného spojenia. *Residuálna* sieť *ResNet-34*, ktorá obsahuje 34 vrstiev konvolučných filtrov, je potom presnejšia a vykazuje podstatne nižšiu chybu pri trénovaní ako *ResNet-18*, ktorá ich obsahuje len 18.

*ResNet-50*, čiže *residuálna* sieť s 50-imi vrstvami, je ďalšou zo sietí postavených na hlbokom *residuálnom* učení. Základný stavebný blok je založený na *bottleneck* architektúre. Oproti klasikej *ResNet-34* je pre každú *residuálnu* funkciu $\mathcal{F}$ použitá naskladaná trojica vrstiev namiesto dvojice (viď. obrázok 3.10 vpravo). Konkrétne sa jedná o $1 \times 1$, $3 \times 3$ a $1 \times 1$ konvolúcie, kde sú $1 \times 1$ vrstvy zodpovedné za zmenšovanie a následne zväčsovanie (obnovenie) dimenzií, zanechávajúc $3 \times 3$ vrstve *bottleneck* s menšími vstupnými a výstupnými dimenziami. Vďaka zavedeniu *bottleneck* architektúry je *ResNet-50* ešte presnejšia ako klasická *ResNet-34*. [13]




Obr. 3.10: Vľavo: Stavebný blok siete *ResNet-34*. Vpravo: *Bottleneck* stavebný blok siete *ResNet-50*. (Obr. prevzatý z [13])

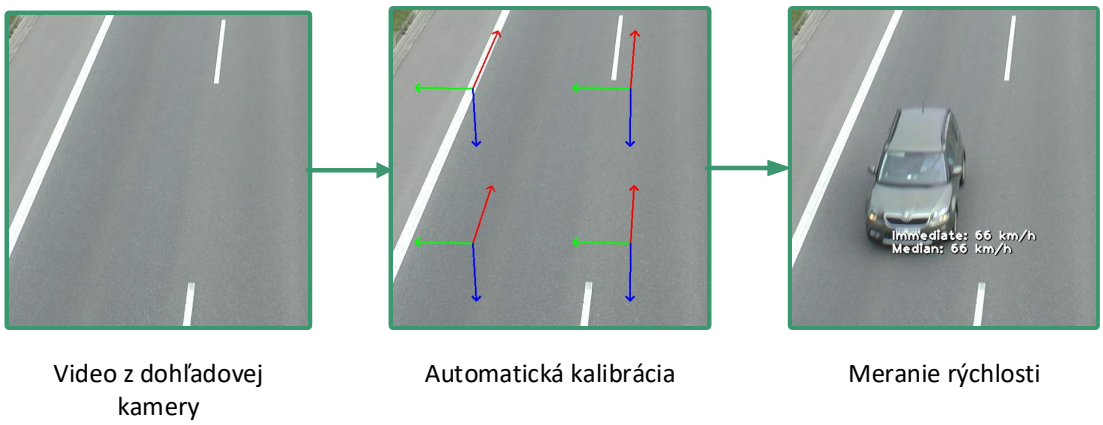
S ešte vyššou presnosťou pracujú siete *ResNet-101* a *ResNet-152*. Tie sú taktiež založené na *bottleneck* architektúre, avšak oproti *ResNet-50* obsahujú väčšie množstvo základných stavebných blokov s tromi vrstvami.

# Kapitola 4

# Návrh riešenia

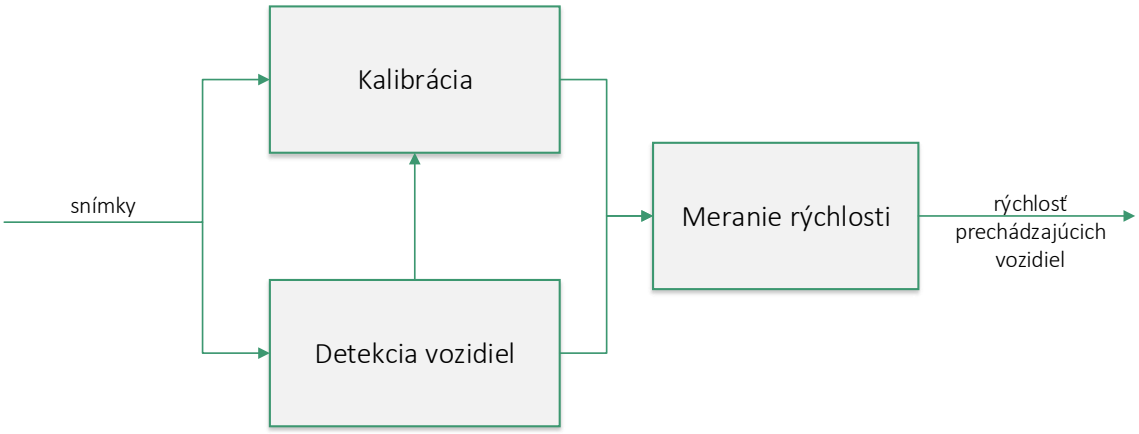
Táto kapitola obsahuje podrobný návrh systému, ktorý má za úlohu riešiť problém, ktorým sa zaoberá táto diplomová práca. Tým je meranie rýchlosti automobilov z dohľadovej kamery. Prvým krokom pri riešení tohto problému je načítanie videa. Tento krok ale nepredstavuje nič neobvyklé či náročné (za použitia knižnice `OpenCV` v programovacom jazyku `Python`, v ktorom bude implementované celé navrhované riešenie), a preto nebude ani ďalej popisovaný. Hlavnou témou tejto kapitoly bude návrh toho, čo sa s načítaným videom bude diať potom.

Činnosť navrhovaného systému je možné po načítaní videa na najvyššej úrovni rozdeliť do dvoch hlavných blokov. Prvý blok predstavuje automatickú kalibráciu a druhý blok samotné meranie rýchlosti prechádzajúcich vozidiel (viď. obrázok 4.1). Blok obsahujúci výpočet rýchlosti bude nadväzovať na blok s kalibráciou, keďže na svoju funkciu potrebuje poznať parametre získané pri kalibrácií.




Obr. 4.1: Hlavné bloky navrhovaného systému

Pri podrobnejšom pohľade je možné na úroveň dvoch hlavných blokov vyčleniť ešte jeden. Týmto blokom je detekcia vozidiel. Nakoľko je potrebná pri niektorých častiach kalibrácie, tak nemôže byť súčasťou podsystému obsahujúceho meranie rýchlosti. Taktiež by bolo ale nevýhodné ju umiestniť do podsystému s automatickou kalibráciou, keďže po dobehnutí samotnej kalibrácie a získaní všetkých potrebných parametrov je tento blok už nepotrebný a zbytočne by bol len kvôli detekcii vozidiel naďalej aktívny. Detekcia vozidiel




Obr. 4.2: Hlavné bloky navrhovaného systému s vyčleneným blokom pre detekciu vozidiel.

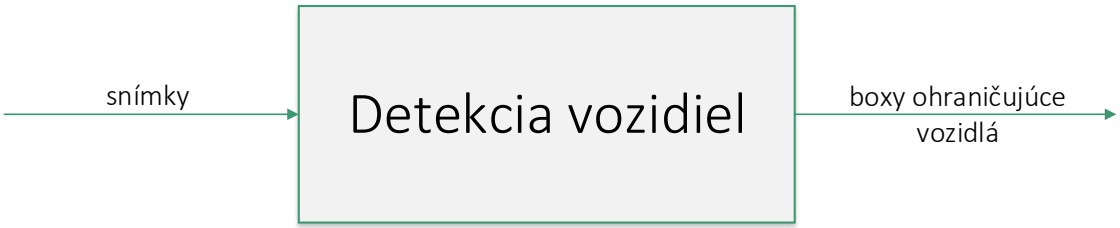
bude teda predstavovať samostatný blok a jej výstup bude posielaný do bloku automatickej kalibrácie ako aj do bloku merania rýchlosti (viď. obrázok 4.2).

Detaily týkajúce sa týchto troch blokov sú podrobnejšie rozpísané v nasledujúcich podkapitolách. Nachádza sa v nich popis ich presného fungovania, schémy aj rozdelenie na ďalšie podsystémy. Podrobnosti ohľadom detekcie vozidiel obsahuje podkapitola 4.1. Navrhovaný podsystém automatickej kalibrácie je popísaný v podkapitole 4.2 a návrh podsystému merania rýchlosti sa nachádza v podkapitole 4.3.

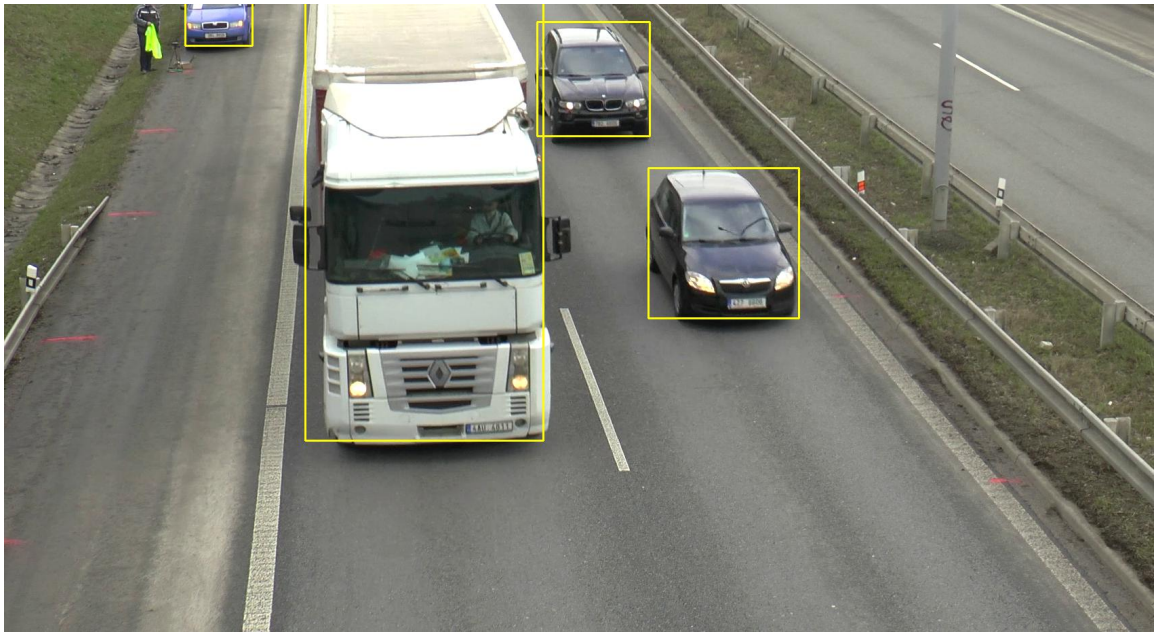
## 4.1 Detekcia vozidiel

Jednoduchá schéma komponentu, ktorý bude zabezpečovať detekciu vozidiel, sa nachádza na obrázku 4.3. Ako je možné vidieť, vstupom budú jednotlivé snímky z videa a na výstupe k nim budú pripojené boxy ohraničujúce vozidlá, ktoré sa podarilo detektoru v daných snímkach detegovať. Vizualizáciu týchto boxov vykreslených do snímky možno vidieť na obrázku 4.4. Na vstupe sa môže nachádzať ešte aj maska na odfiltrovanie vozidiel, ktoré sa nachádzajú v obraze mimo priestoru hlavného záujmu (vozidlá stojace v zábere mimo cestu alebo vozidlá idúce v protismere).

Pri samotnej detekcii bude použitý framework `TensorFlow` [3]. Používať sa bude predtrénovaný detektor uložený ako *frozen inference graph* v *PureBasic* zdrojovom súbore. Tento




Obr. 4.3: Jednoduchá schéma komponentu detektora vozidiel.



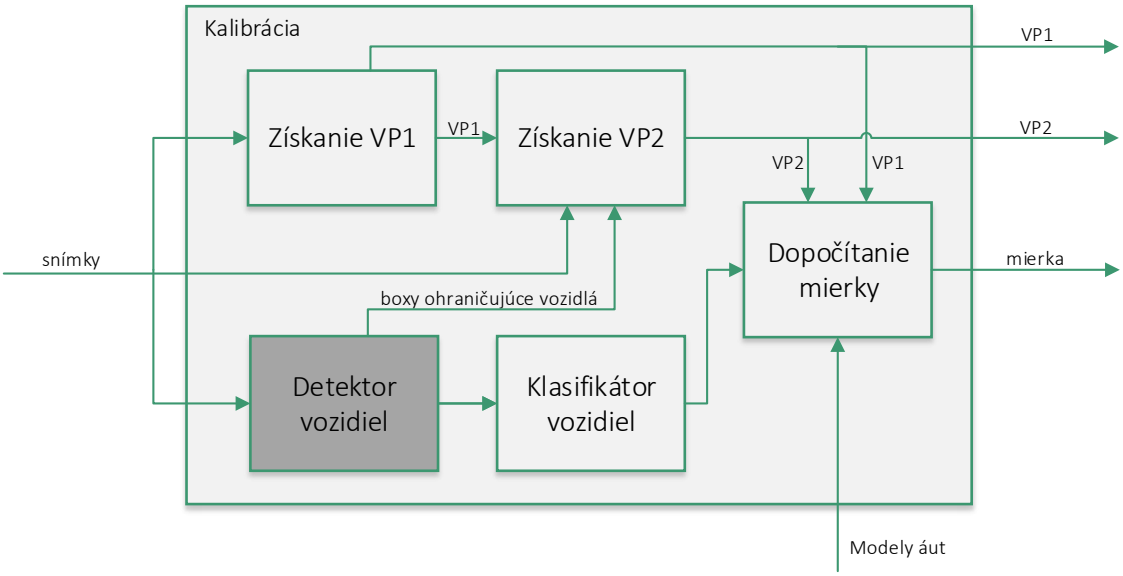
Obr. 4.4: Ukážka boxov ohraničujúcich detegované vozidlá vykreslených do obrazu.

*frozen inference graph* bude pri inicializácii načítaný a následne použitý na detekciu. Systém bude umožňovať použitie akéhokoľvek predtrénovaného detektora, ktorý má používateľ k dispozícii (pri spustení systému bude umožnené zadať cestu k detektoru, ktorý sa má použiť).

V rámci tejto práce bude detekcia vozidiel prebiehať za pomoci moderného konvolučného detektora objektov. Konkrétne sa jedná o detektor využívajúci architektúru *Faster R-CNN* (*Regions with Convolutional Neural Network*) popísaný vyššie v podkapitole 3.3.2. Táto architektúra bola zvolená na základe porovnania obsiahnutého v podkapitole 3.3.3, nakoľko je potrebné, aby bol výstup detektora kvalitný a aby zachytil všetky vozidlá prechádzajúce pred dohľadovou kamerou.

## 4.2 Automatická kalibrácia

Podsystém, ktorý bude vykonávať automatickú kalibráciu, je možné rozdeliť na niekoľko samostatných častí. Ako je možné vidieť na obrázku 4.5, na vstupe tohto podsystému budú snímky načítaného videa a boxy ohraničujúce vozidlá na snímkach (detektor vozidiel je vyznačený tmavšou farbou z dôvodu, že nebude priamo komponentom podsystému kalibrácie). Voliteľný vstup predstavuje maska prekrývajúca oblasti, ktoré sú pre systém nezaujímavé alebo by mohli vniesť do výpočtu nepresnosti. Hlavným výstupom kalibrácie budú prvé dva úbežníky a mierka scény. Okrem toho bude poskytnutý aj demonštratívny grafický výstup zobrazujúci prácu jednotlivých komponentov. Podrobný popis týchto komponentov, z ktorých sa podsystém automatickej kalibrácie skladá, je predmetom nasledujúcich podkapitol.

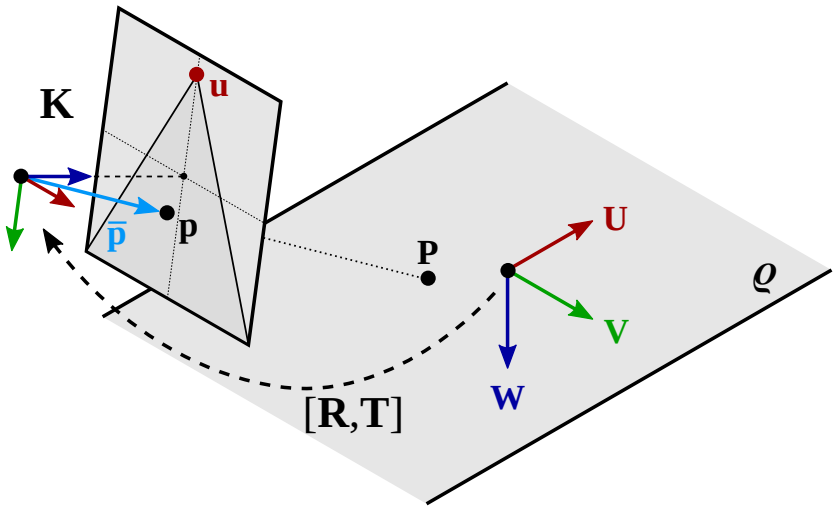



Obr. 4.5: Schéma podsystému zabezpečujúceho automatickú kalibráciu.

### Model dopravnej kamery

Pred samotným návrhom jednotlivých podsystémov je potrebné si zaviesť model dopravnej kamery. Ten bude v tomto riešení vychádzať z modelu použitého v riešení od Sochora a kol. [22]. Predpokladmi pre tento model sú nulové pixelové skosenie a hlavný bod $\mathbf{c}$ v strede obrazu.

Homogénne 2D súradnice obrazu sú reprezentované malými tučnými písmenami $\mathbf{p} = [p_x, p_y, 1]^T$. Body na rovine obrazu v 3D $\overline{\mathbf{p}} = [p_x, p_y, f]^T$, kde $f$ je ohnisková vzdialenosť, sú značené malými tučnými písmenami s vodorovnou čiarkou nad nimi. Ostatné body v 3D (na rovine cesty) sú potom značené veľkými tučnými písmenami $\mathbf{P} = [P_x, P_y, P_z]^T$.




Obr. 4.6: Model a súradný systém kamery. (Obr. prevzatý z [22])

Schému a notáciu tohto modelu je možné vidieť na obrázku 4.6. Rovina cesty je označovaná $\varrho$, prvý úbežník (v smere pohybu vozidiel) je značený ako $\mathbf{u}$, druhý úbežník (kolmý na prvý a paralelný s rovinou cesty) je značený $\mathbf{v}$ a poslený, tretí úbežník (kolmý na rovinu cesty) je označený ako $\mathbf{w}$.

Pomocou prvého úbežníku $\mathbf{u}$, druhého úbežníku $\mathbf{v}$ a za predpokladu hlavného bodu $\mathbf{c}$ v strede obrazu je možné dopočítať ohniskovú vzdialenosť $f$:

$$f = \sqrt{-\mathbf{u}^T \cdot \mathbf{v}},$$

tretí úbežník $\mathbf{w}$ kolmý na rovinu cesty:

$$\begin{aligned}\overline{\mathbf{u}} &= [u_x, u_y, f]^T \\ \overline{\mathbf{v}} &= [v_x, v_y, f]^T \\ \overline{\mathbf{w}} &= \overline{\mathbf{u}} \times \overline{\mathbf{v}},\end{aligned}$$

normalizovaný normálový vektor roviny cesty $\mathbf{n}$:

$$\mathbf{n} = \frac{\overline{\mathbf{w}}}{||\overline{\mathbf{w}}||}$$

a samotnú rovinu cesty $\varrho$:

$$\varrho = [\mathbf{n}^T, \delta]^T.$$

Rovina cesty je však vypočítaná iba v mierke (nakoľko nie je možné zistiť vzdialenosť k rovine cesty iba z úbežníkov), a preto je pridaná ľubovoľná hodnota $\delta = 1$ ako konštantný člen vo vyššie uvedenej rovnici.

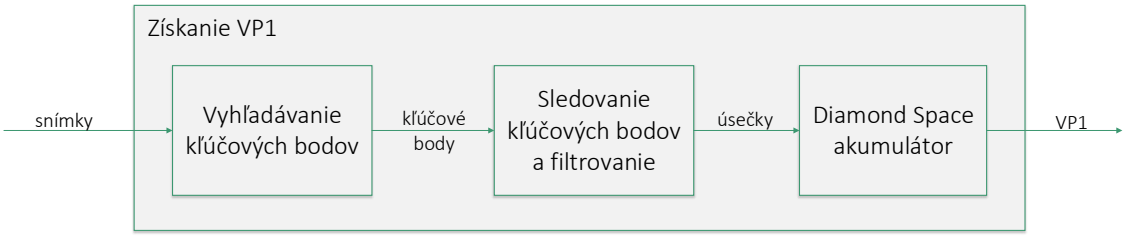
Ak je známa rovina cesty $\varrho$, tak je možné dopočítať 3D súradnice $\mathbf{P} = [P_x, P_y, P_z]^T$ ľubovoľného bodu $\mathbf{p} = [p_x, p_y, 1]^T$ pomocou jeho projekcie na rovinu cesty za použitia nasledujúcich vzorcov:

$$\begin{aligned}\overline{\mathbf{p}} &= [p_x, p_y, f]^T \\ \mathbf{P} &= -\frac{\delta}{[\overline{\mathbf{p}}^T, 0] \cdot \varrho}\overline{\mathbf{p}}\end{aligned}$$

Vzdialenosť na rovine cesty je možné merať priamo pomocou 3D súradníc $\mathbf{P}$. Nakoľko je ale rovina cesty posunutá do preddefinovanej vzdialenosti konštantným členom $\delta = 1$, vzdialenosť $||\mathbf{P}_1 - \mathbf{P}_2||$ medzi bodmi $\mathbf{P}_1$ a $\mathbf{P}_2$ nie je reprezentovaná priamo v metroch (alebo iných jednotkách vzdialenosti skutočného sveta). Je teda potrebné zaviesť ďalší kalibračný parameter značený ako mierka scény $\lambda$. Pomocou tejto mierky je možné konvertovať vzdialenosť $||\mathbf{P}_1 - \mathbf{P}_2||$ z pseudojednotiek roviny cesty na metre škálovaním tejto vzdialenosti na $\lambda||\mathbf{P}_1 - \mathbf{P}_2||$. [22]

### 4.2.1 Získanie prvého úbežníka

Komponent slúžiaci k nájdeniu prvého úbežníku $\mathbf{u}$ bude pozostávať z troch hlavných častí (viď. obrázok 4.7). Nakoľko prvý úbežník $\mathbf{u}$ predstavuje úbežník zodpovedajúci smeru pohybu vozidiel, tak bude pri jeho získavaní využité práve sledovanie pohybu kľúčových bodov detegovaných na vozidlách. Detekcia týchto kľúčových bodov predstavuje teda prvú časť komponentu. Detegované budú v snímkach, ktoré sú na vstupe (okrem nich môže byť na vstupe aj maska použitá na filtráciu kľúčových bodov). Na ich detekciu bude použitá knižnica `OpenCV`, ktorá dokáže jednoducho vyhľadať body vhodné na sledovanie v obraze. Ukážka takto vyhľadaných bodov sa nachádza na obrázku 4.8 vľavo.

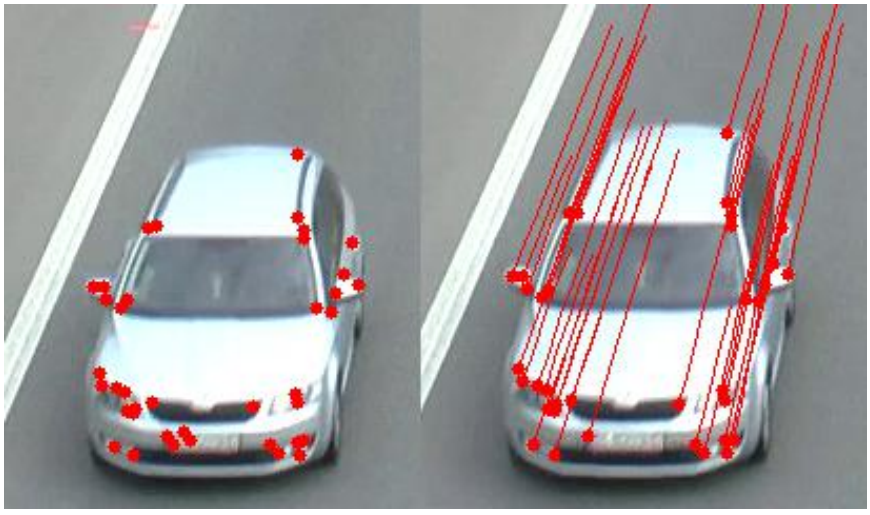



Obr. 4.7: Schéma podsystému zabezpečujúceho určenie prvého úbežníka.

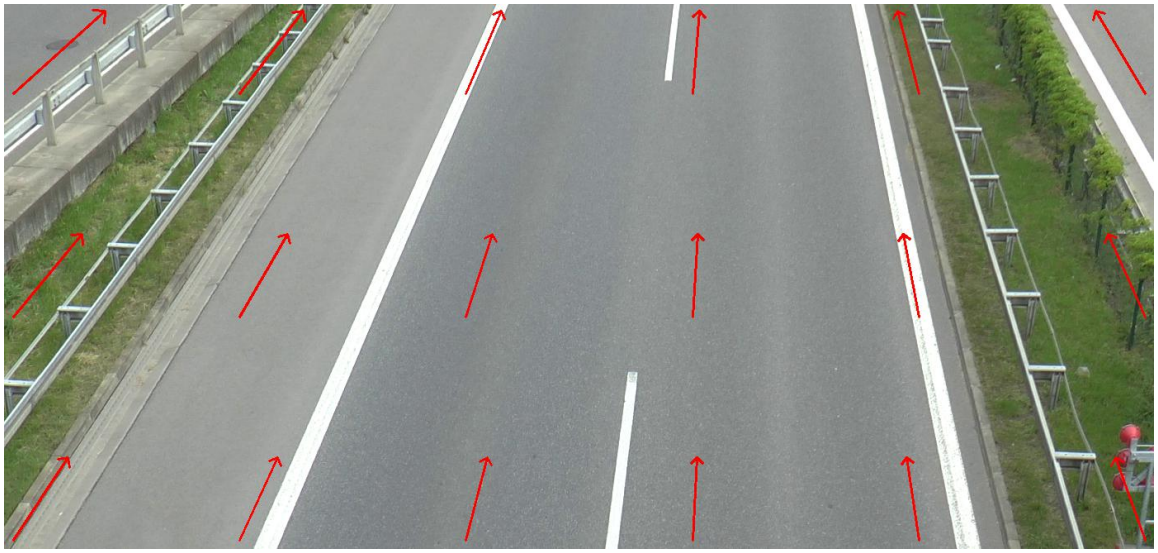
Tieto kľúčové body sú následne sledované pomocou metódy *Lucas-Kanade* sledovania optického toku. Táto metóda bude taktiež implementovaná pomocou knižnice `OpenCV`. Kľúčové body, ktoré sa medzi snímkami nepohybujú (nie sú teda súčasťou prechádzajúcich vozidiel, ale prostredia), budú odfiltrované, aby nespôsobovali chybné výsledky. Pohyb sledovaných kľúčových bodov bude zaznamenávaný ukladaním ich $x$-ových a $y$-ových súradníc naprieč snímkami. Vizuálna reprezentácia úsečiek, ktoré tieto body vytvárajú, sa nachádza na obrázku 4.8 vpravo. Súradnice týchto bodov budú priebežne zasielané na vstup *Diamond Space* akumulátora vždy, keď budú reprezentovať 2.500 nových úsečiek.

Na vstupe poslednej časti komponentu slúžiaceho na získanie prvého úbežníku budú tieto úsečky premenené na priamky a naakumulované do *Diamod Space*-u, ktorý (spôsobom popísaným bližšie v podkapitole 3.1.1) následne vráti hľadaný úbežník $\mathbf{u}$. Úbežník bude získavaný každých 2.500 priamok z dôvodu jeho použitia v ďalšom komponente, ktorý vďaka jeho približnej polohe bude môcť skôr začať svoj výpočet. Beh celého komponentu bude ukončený po naakumulovaní 32.500 priamok, keďže presnosť polohy prvého úbežníku $\mathbf{u}$ je pri danom počte priamok už dostatočná.

Priebeh výpočtu komponentu slúžiaceho na získanie prvého úbežníku bude demonštratívne zobrazovaný v grafickom výstupe. Tento výstup predstavujú po sebe idúce snímky (video), do ktorých sú vykreslované kľúčové body a trasy ich pohybu (viď. obrázok 4.8



Obr. 4.8: Vykreslené detegované kľúčové body vhodné na sledovanie sa nachádzajú vľavo. Vykreslené úsečky reprezentujúce pohyb týchto kľúčových bodov voči predchádzajúcim snímkam sa nachádzajú vpravo.



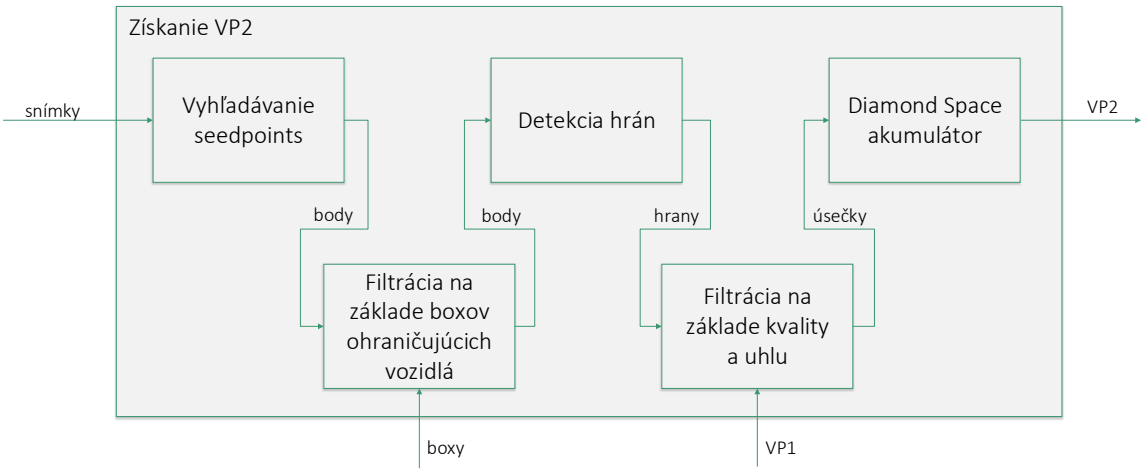
Obr. 4.9: Demonštratívny grafický výstup komponentu s vykreslenými šípkami smerujúcimi k prvému úbežníku **u**.

vpravo) ako aj šípky smerujúce k priebežne dopočítavanému prvému úbežníku **u** (viď. obrázok 4.9).

### 4.2.2 Získanie druhého úbežníka

Získanie druhého úbežníku **v**, kolmého na smer pohybujúcich sa vozidiel a paralelného s rovinou cesty $\varrho$, bude náročnejšie a komplexnejšie ako získanie prvého úbežníku. Podsystém na jeho získanie je možné rozdeliť do piatich hlavných blokov, ktoré je vidno na obrázku 4.10.

Na vstupe tohto podsystému tak nebudú len samotné snímky z načítaného videa, ale aj predtým získaný prvý úbežník a boxy ohraničujúce vozidlá na snímkach získané pomocou detektora vozidiel. Tento komponent teda môže zahájiť svoj výpočet až po prvom priebež-




Obr. 4.10: Schéma podsystému zabezpečujúceho určenie druhého úbežníka.

nom orientačnom výpočte predchádzajúceho komponentu a môže spracovávať len snímky, ktoré už spracoval detektor vozidiel. Snímky na vstupe tohto komponentu sú najprv zmenšené na jednu tretinu svojej veľkosti, aby sa zvýraznili silné hrany v obraze.

Následne prichádza na rad prvá významná časť komponentu, a to vyhľadávanie *seedpoints*. *Seedpoints* sú získané ako lokálne maximá veľkostí gradientu obrazu. Tieto body majú vysokú pravdepodobnosť, že sa nachádzajú na nejakej hrane v obraze (viď. zelené body na obrázku 4.11 úplne vľavo). Pred detegovaním hrán ale budú tieto body prefiltrované pomocou boxov ohraničujúcich vozidlá a body nachádzajúce sa mimo týchto boxov budú odstránené. Táto filtrácia bude zavedená z dôvodu, že sa tieto hrany, na ktorých sa filtrované body nachádzajú, vyskytujú v každej snímke (hrany v prostredí) a mohli by značne ovplyvniť výsledok a zaviesť do výpočtu chybu.

Z odfiltrovaných bodov budú detegované hrany nasledujúcim spôsobom. Pre každý *seedpoint* $s = [x, y]$ je zostavená matica $X$ z okolia o rozmere $9 \times 9$:

$$X = \begin{bmatrix} w_1(m_1 - x) & w_1(n_1 - y) \\ w_2(m_2 - x) & w_2(n_2 - y) \\ \vdots & \vdots \\ w_k(m_k - x) & w_k(n_k - y) \end{bmatrix},$$

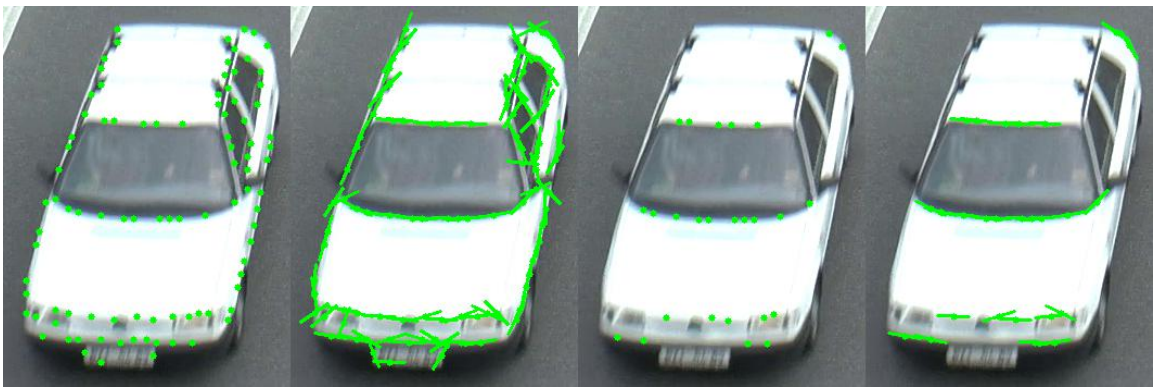
kde $[m_k, n_k]$ sú súradnice okolitých pixelov ($k = 1...81$) a $w_k$ predstavuje veľkosť gradientu daného pixelu z okolia. Pre okolie o rozmere $9 \times 9$ teda vznikne matica $X$ o rozmere $81 \times 2$. Vlastné vektory a čísla matice $X^T X$ potom možno dopočítať nasledovne:

$$W \Sigma^2 W^T = \mathrm{SVD}(X^T X),$$

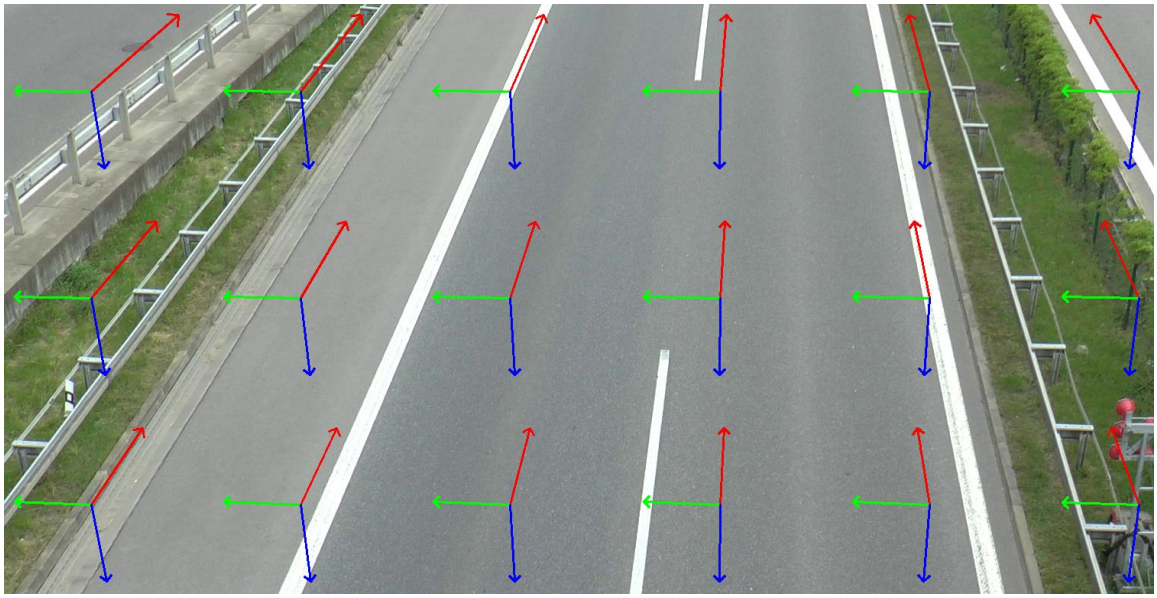
kde

$$W = [a_1, a_2]$$
$$\Sigma = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

Vektory $a_1$ a $a_2$ predstavujú vlastné vektory matice $X$, zatiaľ čo $\lambda_1$ a $\lambda_2$ označujú korešpondujúce vlastné čísla. Orientáciu hrany potom predstavuje vlastný vektor $a_1$ a kvalitu hrany je možné vyjadriť pomerom $\frac{\lambda_1}{\lambda_2}$. [22]



Obr. 4.11: Vizualizácia detekcie a filtrácie hrán. Zľava doprava – *Seedpoints* ako lokálne maximá veľkostí gradientu obrazu; Detegované hrany, na ktorých dané *seedpoints* ležia; *Seedpoints* ležiace na hranách, ktoré ostali po filtrácii; Hrany, ktoré ostali po filtrácii na základe uhlu a kvality.



Obr. 4.12: Demonštratívny grafický výstup komponentu s vykreslenými šípkami smerujúcimi k prvému úbežníku **u** (červená farba), druhému úbežníku **v** (zelená farba) a tretiemu úbežníku **w** (modrá farba) tvoriacimi súradný systém snímanej situácie.

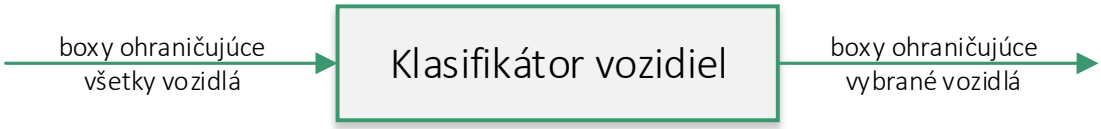
Po získaní hrán (s ich orientáciou a kvalitou) nasleduje ich filtrácia. Filtrované sú podľa ich kvality aj podľa orientácie. Tie, ktoré majú kvalitu nižšiu ako 3 alebo ich odklon od smeru k prvému úbežníku **u** alebo odhadnutému tretiemu úbežníku **w** je menší ako 25°, sú odfiltrované. Zvyšné hrany (viď. obrázok 4.11 úplne vpravo) sú prevedené na úsečky, ktoré sú po naakumulovaní 2.500 kusov ďalej zaslané do *Diamod Space* akumulátora.

Priestor akumulátora ale musí byť ešte pred dopočítaním druhého úbežníku **v** vymaskovaný na základe očakávaného rozsahu ohniskovej vzdialenosti a uhlu horizontu. Preto musí byť pred dopočítaním finálnej polohy druhého úbežníku **v** známa finálna poloha prvého úbežníku **u**. Ak je výpočet prvého úbežníku **u** dokončený a súčasne je naakumulovaných 7.500 úsečiek reprezentujúcich detegované hrany, tak prebehne záverečný výpočet druhého úbežníku **v**. Ten je následne predaný na vstup ďalšieho podsystému, ktorý ho potrebuje na svoje výpočty.

Rovnako ako predchádzajúci komponent, tak aj komponent na získanie druhého ubežníku **v** poskytuje aj grafický výstup slúžiaci na demonštrovanie jeho činnosti. Ten je zobrazovaný po ukončení zobrazovania grafického výstupu predchádzajúceho komponentu a sú doňho vykreslované odfiltrované hrany so *seedpoints*, z ktorých boli detegované (viď. obrázok 4.11 úplne vpravo), a šípky smerujúce k priebežne dopočítavanému druhému úbežníku **v**. Okrem šípok smerujúcich k druhému úbežníku **v** sa do grafického výstupu vykreslujú aj šípky smerujúce k prvému úbežníku **u** a dopočítanému tretiemu úbežník **w**. Tieto trojice šípok tak znázorňujú súradný systém snímanej situácie (viď. obrázok 4.12).

### 4.2.3 Klasifikátor vozidiel

Klasifikácia určitých modelov vozidiel je potrebná pre získanie mierky. Vstup predstavuje snímka s boxami ohraničujúcimi vozidlá, ktoré boli na nej detegované (viď. obrázok 4.13). Samotná klasifikácia bude implementovaná za použitia knižnice `Keras` [2]. Pomocou nej bola aj predtrénovaná sieť slúžiaca na klasifikáciu. Tá bola natrénovaná nad takmer 113.000




Obr. 4.13: Jednoduchá schéma komponentu zabezpečujúceho klasifikáciu vozidiel.

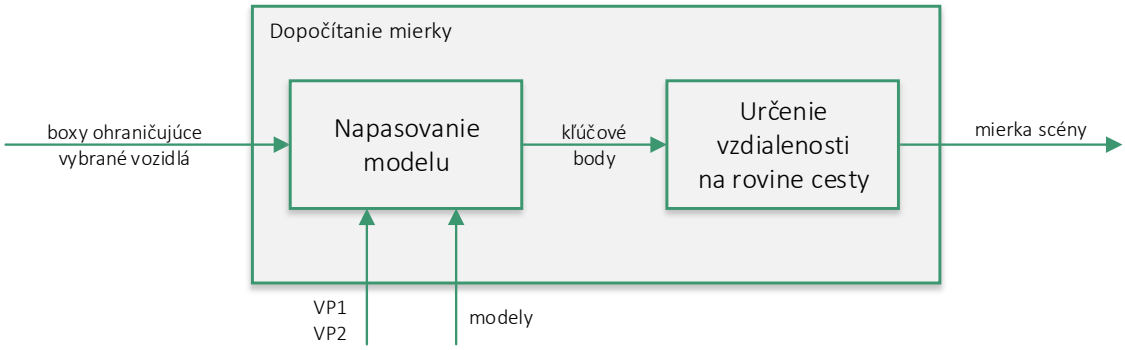
obrázkami vozidiel z datasetu BoxCars116k [24]. Klasifikátor je natrénovaný rozpoznávať 8 tried automobilov, a to: *Škoda Fabia I combi*, *Škoda Fabia II combi*, *Škoda Fabia II hatchback*, *Škoda Octavia I combi*, *Škoda Octavia II combi*, *Škoda Octavia I sedan*, *Škoda Octavia II sedan* a *ostatné*. Pri trénovaní bol využitý predimplementovaný model siete *ResNet50* (podrobnejší popis princípu jeho fungovania sa nachádza v podkapitole 3.4.2), ktorý obsahuje knižnica `Keras`. Táto knižnica taktiež umožňuje uložiť natrénovanú sieť vo formáte `H5` (*Hierarchical Data Format 5*) a znovu ju načítať a používať.

Komponent slúžiaci na klasifikáciu teda načíta predtrénovanú sieť, pričom je znovu umožnené pri spúšťaní systému určiť, aký klasifikátor sa použije. Používateľom je teda daná možnosť použiť akýkoľvek iný, im dostupný predtrénovaný klasifikátor uložený vo formáte `H5`. Zo snímky budú následne vyrezané pomocou ohraničujúcich boxov detegované autá. Tie budú vyhodnotené klasifikátorom, a pokiaľ je im priradená iná trieda ako *ostatné*, tak komponent pošle daný box spolu so zodpovedajúcou triedou na výstup (viď. obrázok 4.13).

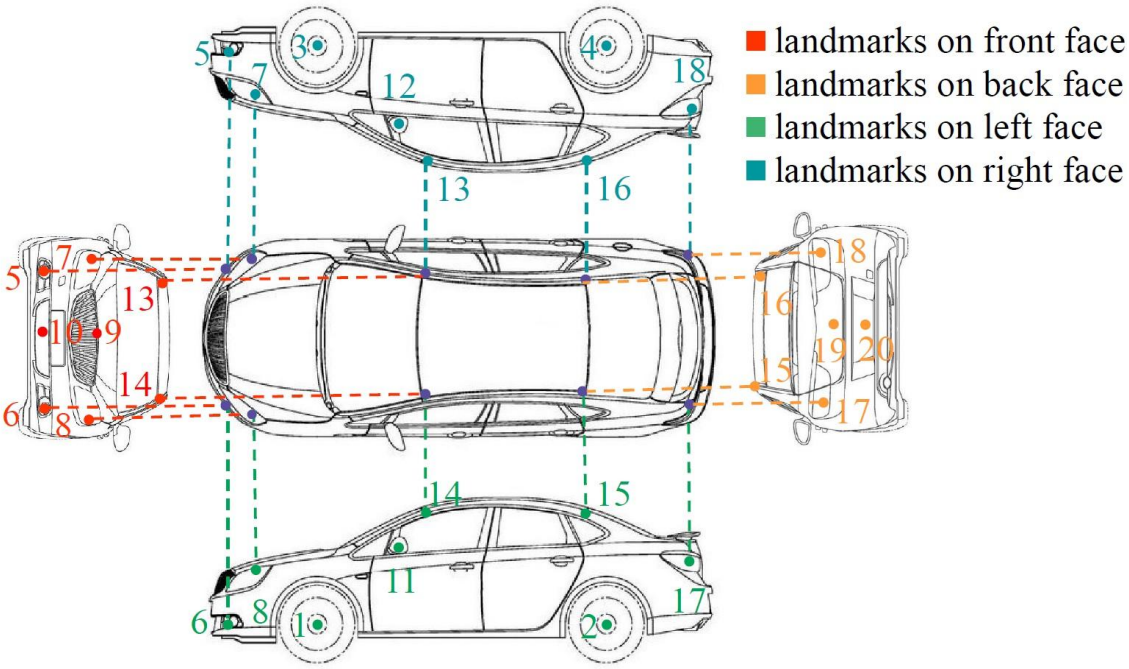
### 4.2.4 Dopočítanie mierky

Dopočítanie mierky predstavuje posledný komponent podsystému automatickej kalibrácie. Jeho účelom je zistiť mierku scény $\lambda$, pomocou ktorej je možné konvertovať pseudojednotky roviny cesty na metre. Jednoduchá schéma procesu dopočítania mierky scény je zobrazená na obrázku 4.14.

Ako je možné v tejto schéme vidieť, prvým krokom procesu získania mierky bude napasovávanie vybraných modelov na detegované autá v obraze. Vstupom tejto časti sú teda boxy ohraničujúce vybrané vozidlá z komponentu klasifikátora vozidiel, 3D modely áut zodpove-




Obr. 4.14: Schéma komponentu zabezpečujúceho dopočítanie mierky scény.



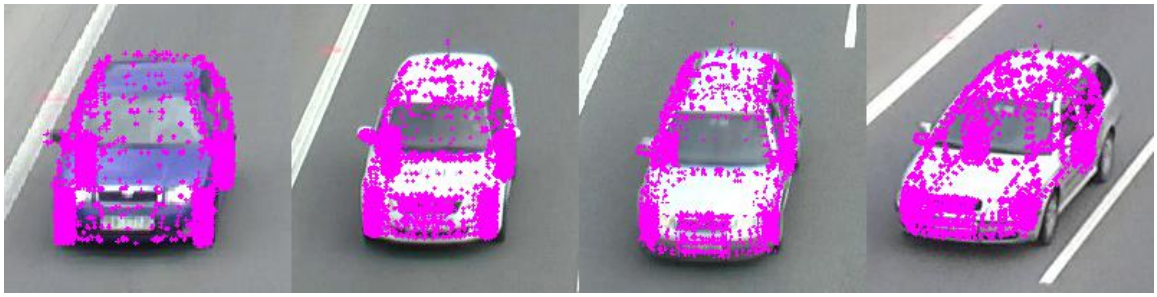

Obr. 4.15: Kľúčové body na vozidle použité pri získavaní mierky. (Obr. prevzatý z [27])

dajúce tým, ktoré má za úlohu klasifikátor rozpoznať, prvý úbežník **u** a druhý úbežník **v**. Z tohto dôvodu je tento komponent spúšťaný až po dobehnutí výpočtu v komponentoch na získanie prvých dvoch úbežníkov.

Spomínané modely budú načítané pri inicializácii z priečinku, do ktorého komponent dostane zadanú cestu. Modely sú uložené vo formáte `CSV` (*Comma-Separated Values*). Konkrétne sú v súboroch vyexportované 3D body ležiace na danom modeli auta. Súradnice týchto bodov zodpovedajú reálnym rozmerom a vzdialenosti medzi nimi sú v metroch. Okrem vyexportovaných modelov sú v osobitných súboroch uložené aj kľúčové body z týchto modelov. Znázornenie ich polohy na vozidle možno vidieť na obrázku 4.15.

Vyexportované 3D modely budú napasované do ohraničujúcich boxov na vstupe. Prvým krokom je zostavenie rotačnej matice $R$ z úbežníkov v bode reprezentujúceho polohu auta v obraze (konkrétne bude použitý bod nachádzajúci sa uprostred ohraničujúceho boxu). Zodpovedajúci model bude následne pomocou tejto matice zrotovaný a dochádza k výpočtu koeficientu $k$, ktorým je potrebné body zrotovaného modelu prenásobiť, aby dosiahol veľkosť detegovaného vozidla. Tento koeficient bude dopočítaný ako pomer výšky ohraničujúceho boxu a výšky zrotovaného modelu. Konkrétne sa výška modelu dopočíta z najnižšiej $y$-ovej polohy a druhej najvyššej $y$-ovej polohy bodov nachádzajúcich sa v modeli. Najvyššia $y$-ová poloha nebude braná v úvahu z dôvodu, že sa na tom mieste často nachádza bod reprezentujúci anténu, ktorá nebýva súčasťou ohraničujúceho boxu. Po získaní hľadaného koeficientu $k$ bude model zväčšený a na základe polohy detegovaného vozidla v obraze posunutý na správnu pozíciu. Na túto pozíciu je aj vykreslený pre demonštratívne účely (viď. obrázok 4.16).

Pomocou rotačnej matice $R$, koeficientu $k$ a počiatku, do ktorého je potrebné model posunúť v rámci obrazu, bude následne zrotovaný, zväčšený a posunutý zodpovedajúci model s kľúčovými bodmi. Z upraveného modelu bude dopočítaná vzdialenosť kľúčových bodov na rovine cesty. Pri výpočte budú konkrétne použité body 1 a 2 (viď. obrázok 4.15), ktoré



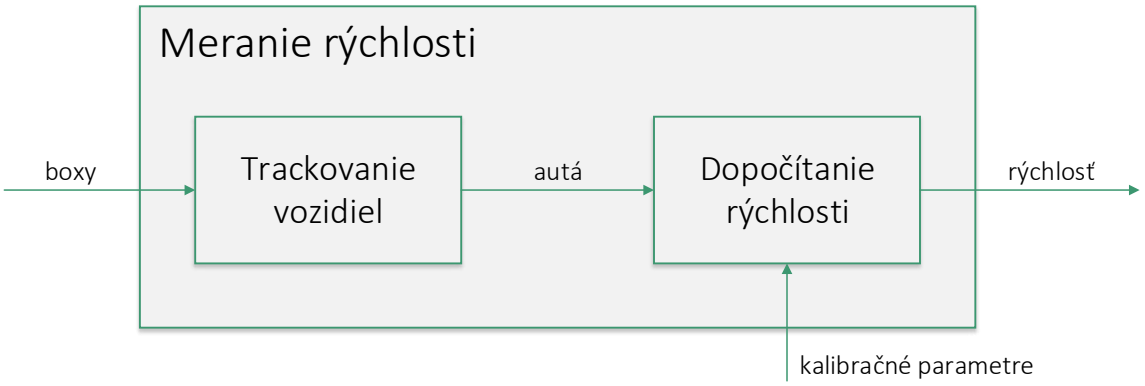
Obr. 4.16: Vizualizácie zrotovaného, zväčšeného a posunutého modelu na mieste zodpovedajúcom výskytu vozidla.

ležia najbližšie k rovine cesty. Reálnu vzdialenosť týchto bodov v metroch predstavuje ich vzdialenosť z pôvodného modelu s klúčovými bodmi. Mierku je následne možné dopočítať ako podiel reálnej vzdialenosti a vzdialenosti na rovine cesty. Jednotlivé merania budú ukladané a výsledná mierka scény $\lambda$ bude definovaná po 186 meraniach ako medián získaných mierok.

Získaním mierky scény $\lambda$ je teda výstup podsystému automatickej kalibrácie, spolu s prvým úbežníkom $\mathbf{u}$ a druhým úbežníkom $\mathbf{v}$, kompletný. Systém teda disponuje všetkými kalibračnými parametrami potrebnými na spustenie posledného z hlavných podsystémov a začatie merania rýchlosti prechádzajúcich vozidiel.

## 4.3 Meranie rýchlosti

Samotný podsystém merania rýchlosti sa dá rozdeliť do dvoch na seba nadväzujúcich častí. Ako je možné vidieť na obrázku 4.17, prvú časť predstavuje *trackovanie* vozidiel a druhú dopočítavanie rýchlosti týchto vozidiel. Vstupom celého podsystému sú tak ohraničujúce boxy detegovaných vozidiel v jednotlivých snímkach (výstup podsystému detekcie vozidiel) a kalibračné parametre špecifikujúce snímanú situáciu (výstup podsystému automatickej kalibrácie). Obe časti podsystému merania rýchlosti sú bližšie popísané v nasledujúcich podkapitolách.




Obr. 4.17: Schéma podsystému zabezpečujúceho meranie rýchlosti prechádzajúcich vozidiel.

### 4.3.1 Trackovanie vozidiel

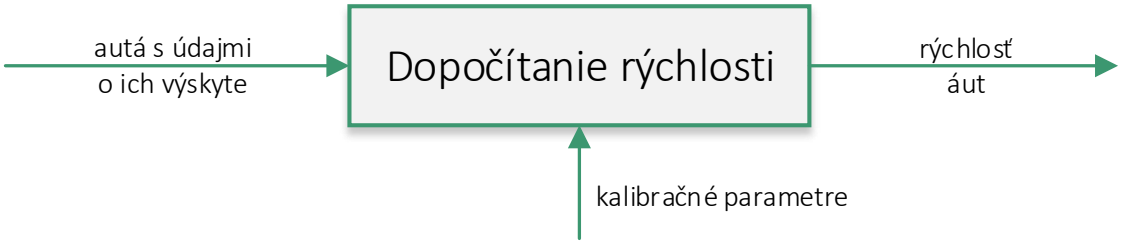

Obr. 4.18: Jednoduchá schéma komponentu zabezpečujúceho trackovanie prechádzajúcich vozidiel.

*Trackovanie* vozidiel má za účel zmapovať pohyb jednotlivých áut detegovaných v obraze. Aby bolo možné dopočítať ich rýchlosť, je potrebné vedieť, kde presne sa v ktorej snímke nachádzali. Na vstupe tohto komponentu sa budú nachádzať boxy ohraničujúce vozidlá (viď. obrázok 4.18), ktoré boli detegované pomocou detektora vozidiel. Tieto boxy budú spracované tak, že budú porovnané s boxami z predchádzajúcich snímok. Konkrétne bude porovnávaný bod ležiaci v strede spodnej hrany ohraničujúceho boxu. Pokiaľ bude tento bod od bodu z predchádzajúcej snímky vzdialený v rámci určitého limitu smerom od, resp. k prvému úbežníku, tak bude táto pozícia s číslom snímky priradená k danému autu z predchádzajúcej snímky. Pokiaľ nebudú pre niektoré pozície nájdené ekvivalenty v predchádzajúcej snímke, tak bude pre zhodu prehľadávaná snímka pred ňou, až po limit 6 snímok dozadu. Pokiaľ ani v týchto snímkach nebude nájdený box, ku ktorému by bolo možné spracovávaný box priradiť, tak bude daný box vyhodnotený ako nové auto. Výstupom tohto komponentu sú následne autá s údajmi o ich pozíciách a číslach snímok, v ktorých boli na danej pozícii nájdené (viď. obrázok 4.18).

### 4.3.2 Dopočítanie rýchlosti

Posledným komponentom pred hlavným výstupom celého systému je komponent zabezpečujúci dopočítanie rýchlosti na základe kalibračných parametrov a údajov o výskyte áut (viď. orázok 4.19).

Pri samotnom výpočte rýchlosti bude navrhovaný systém taktiež vychádzať z riešenia od Sochora a kol. [22]. Údaje o výskyte áut na vstupe obsahujú pre každé auto referenčné body




Obr. 4.19: Jednoduchá schéma komponentu zabezpečujúceho dopočítanie rýchlosti prechádzajúcich vozidiel.

$\mathbf{p}_i$ (bod uprostred spodnej hrany ohraničujúceho boxu) spolu s číslami snímok, v ktorých sa dané auto nachádzalo v týchto bodoch. Z údaju o čísle snímky sa dá získať časová značka $t_i$ pre každý referenčný bod. Na jeho získanie je potrebné poznať údaj o počte snímok za sekundu. Ten bude zadávaný používateľom pri spustení, nakoľko jeho automatické zisťovanie pomocou vlastnosti `CAP_PROP_FPS` triedy `VideoCapture` v `OpenCV` sa preukázalo ako nepresné.

Rýchlosť $v$ je potom dopočítaná pomocou nasledujúceho vzorca:

$$v = \underset{i=1\ldots N-\tau}{\text{median}} \left( \frac{\lambda ||\mathbf{P}_{i+\tau} - \mathbf{P}_i||}{t_{i+\tau} - t_i} \right),$$

pričom body $\mathbf{P}_i$ predstavujú referenčné body $\mathbf{p}_i$ premietnuté na rovinu cesty (princíp tejto projekcie je popísaný vyššie v podkapitole 4.2) a $\lambda$ predstavuje mierku scény. Jednotlivé rýchlosti sú vypočítané medzi po sebe idúcimi pozíciami v čase a výsledná rýchlosť potom predstavuje ich medián. Konštanta $\tau$ je zavedená pre vyššiu stabilitu merania. Rýchlosť je totiž presnejšia, pokiaľ nie sú použité pozície zo snímok idúcich hneď po sebe, ale pokiaľ je medzi danými snímkami určitý rozostup. V navrhovanom systéme bude použitá konštanta $\tau = 4$.

Tento komponent bude taktiež zobrazovať demonštratívny grafický výstup. Jeho príklad možno vidieť na obrázku 4.20. Do zodpovedajúcich snímok je k vozidlám vpísaná ich okamžitá (*Immediate*) a výsledná (*Median*) rýchlosť. Výpočet výslednej rýchlosti je popísaný vyššie, jej hodnota sa ale taktiež s postupom vozidla mení, keďže sa akumulujú rýchlosti, z ktorých je získavaný medián. Okamžitá rýchlosť predstavuje rýchlosť dopočítanú medzi polohou vozidla v aktuálnej snímke a jeho polohou 4 snímky dozadu.

Týmto komponentom bude uzatvorený celý systém a jeho výstup bude hlavným výstupom celého navrhovaného systému pre meranie rýchlosti automobilov z dohľadovej kamery. Okrem demonštratívneho grafického výstupu bude generovaný výstup vo formáte `JSON`


Obr. 4.20: Ukážka demonštratívneho grafického výstupu zobrazujúceho nameranú rýchlosť prechádzajúcich vozidiel.

ukladaný do súboru, ktorý bude obsahovať získané údaje ku všetkým detegovaným vozidlám, ktoré prešli pred dohľadovou kamerou. Okrem toho budú do výstupného súboru vypísané aj kalibračné parametre ($\mathbf{u}$, $\mathbf{v}$, $\mathbf{c}$, $\lambda$) získané podsystémom automatickej kalibrácie.

# Kapitola 5

# Implementácia

Táto kapitola obsahuje detaily týkajúce sa implementácie systému navrhnutého v kapitole 4. Ako programovací jazyk, v ktorom je výsledný systém naimplementovaný, bol zvolený jazyk `Python`, a to z dôvodu ľahkej integrácie mnohých knižníc a frameworkov, ktoré sa používajú na spracovanie obrazu (napr. `OpenCV`, `NumPy`, `SciPy`, `TensorFlow`, `Keras` ...).

Navrhnutý systém je naimplementovaný ako uzatvorený celok, ktorý užívateľ len spustí so zvolenými vstupnými parametrami. Predávanie medzivýsledkov, uchovávanie potrebných hodnôt, spúšťanie jednotlivých podsystémov a komponentov vykonáva systém sám. Výhodou takéhoto prístupu je jednoduchšie použitie systému ako celku. Nevýhodou je ale vysoká náročnosť na pamäť, keďže si systém uchováva všetky snímky a potrebné dáta až do doby, kým je spracovanie danej snímky ukončené jej zobrazením v rámci podsystému merania rýchlosti. Aj z tohto dôvodu je podsystém automatickej kalibrácie optimalizovaný tak, aby nemusel načítať všetky snímky z videa. Celé video je tak načítané až v rámci podsystému merania rýchlosti, kde už ale jednotlivé snímky nie sú uchovávané po tak dlhú dobu ako v podsystéme automatickej kalibrácie. Snímky sú v rámci systému uchovávané ako inštancie triedy `Frame`. Tá obsahuje okrem načítanej snímky (obrázku) jej číslo, vlastnosti ako šírku, výšku, hlavný bod a po spracovaní detektorom aj boxy ohraničujúce vozidlá, ktoré v nej boli detegované.

Systém je implementovaný tak, že jednotlivé podsystémy aj komponenty sú spúšťané v samostatných vláknach. To umožňuje využitie paralelizmu pri výpočte, z ktorého plynie jeho zrýchlenie. Taktiež sa tým ale zvyšuje požiadavka na výpočetný výkon potrebný pre plynulý beh systému. Optimalizáciu predstavuje ukončovanie výpočtu jednotlivých komponentov a podsystémov okamžite po dosiahnutí dostatočnej presnosti ich výsledku. Toto je uplatňované hlavne v rámci podsystému automatickej kalibrácie, v ktorom jednotlivé komponenty nespracuvávajú celé video, ale len jeho úsek. Predávanie inštancií triedy `Frame` medzi jednotlivými komponentami bežiacimi vo vláknach je vyriešené pomocou frontov.

Postup pri spúšťaní systému je predmetom prílohy B a príloha C obsahuje informácie ohľadom systémových požiadaviek, ktoré je potrebné pre spustenie naimplementovaného systému splniť.

### Výstup systému

Hlavným výstupom systému je súbor so získanými údajmi. Konkrétne sa jedná o výsledné hodnoty kalibračných parametrov (prvý úbežník $\mathbf{u}$ – `vp1`, druhý úbežník $\mathbf{v}$ – `vp2`, hlavný bod $\mathbf{c}$ – `pp` a mierka scény $\lambda$ – `scale`) a informácie o trackovaných vozidlách. Štruktúra

výstupného súboru vo formáte JSON je rovnaká, ako sa používa v datasete *BrnoCompSpeed* [23] (z dôvodu vyhodnocovania presnosti), a to nasledovná:

```
{
	"camera_calibration":
		{
			"vp1": [x,y],
			"vp2": [x,y],
			"pp": [x,y],
			"scale": lambda
		},
	"cars":
		[{
			"id": number,
			"frames": [numbers],
			"posX": [numbers],
			"posY": [numbers]
		}]
	}
}
```

V prípade, že si užívateľ zvolí pri spustení, že sa má vypisovať aj rýchlosť, tak je do výstupného súboru vypísaná ku každému autu aj výsledná nameraná rýchlosť (do štruktúry je pridaný kľúč `"speed"`). Samotné vytvorenie štruktúry výstupného súboru je vďaka internej reprezentácii trackovaných áut jednoduché. Každé auto predstavuje inštanciu triedy `Car`, ktorá obsahuje vlastnosti zhodné s kľúčmi výstupného súboru.

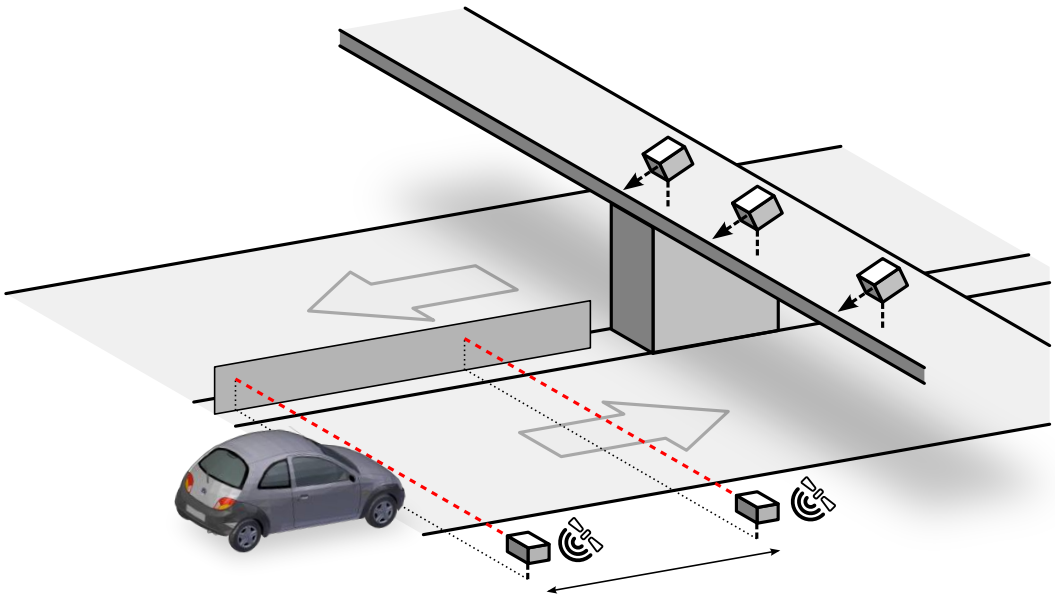
# Kapitola 6

# Testovanie a vyhodnotenie

V tejto kapitole je popísané, akým spôsobom a na akých dátach bol implementovaný systém testovaný. Okrem toho táto kapitola obsahuje výsledky testovania a vyhodnotenie úspešnosti jednotlivých častí celého riešenia. Systém bude taktiež porovnávaný s výsledkami iných obdobných systémov, ktoré riešia problém tejto práce.

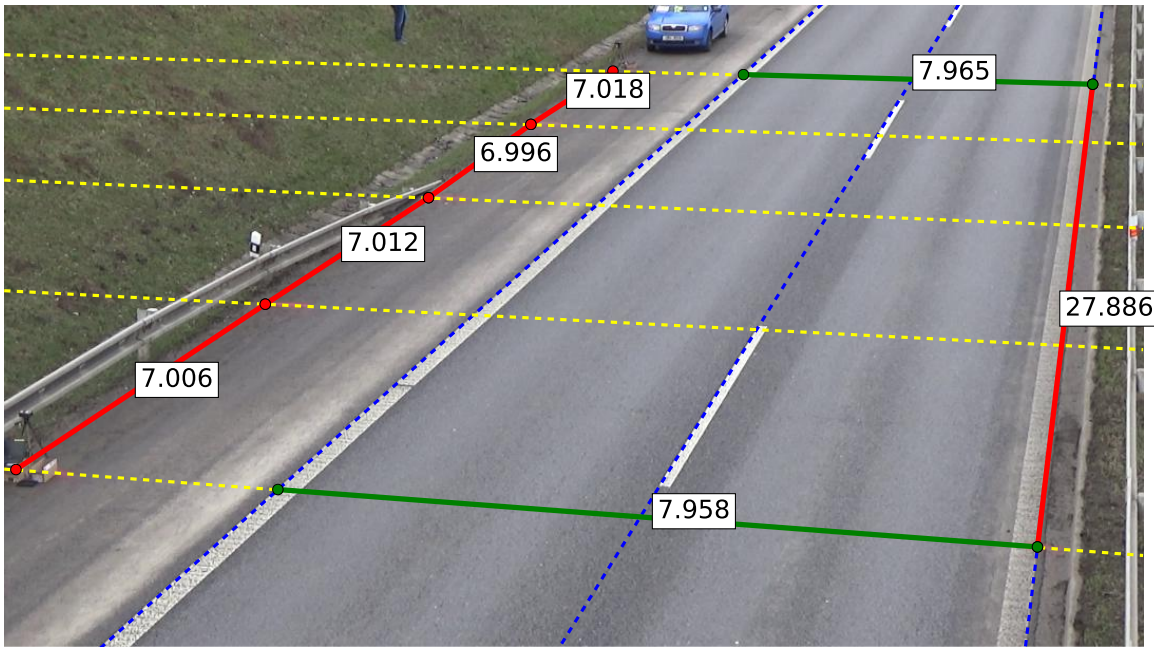
## 6.1 Testovacia sada

Testovaciu sadu, za použitia ktorej bol systém vyvíjaný, ladený a následne testovaný, predstavuje dataset *BrnoCompSpeed* [23]. Tento dataset obsahuje 18 full-HD videí. Každé z nich má približne hodinu a boli nasnímané v šiestich rôznych lokalitách. Na každú lokalitu tak pripadajú tri videá snímané z kamier na rozličných pozíciách (`cam-right`, `cam-center` a `cam-left` na obrázku 6.1).



Obr. 6.1: Schéma nahrávania a získavania dát z datasetu *BrnoCompSpeed*. (Obr. prevzatý z [23])

Tento dataset ďalej obsahuje anotáciu všetkých vozidiel vyskytujúcich sa vo videách s presnými meraniami ich rýchlosti. Tie boli získané z optických brán za použitia metódy LIDAR a overené niekoľkými referenčnými GPS stopami. Vďaka týmto informáciam je vhodný na testovanie a následné vyhodnocovanie presnosti implementovaného systému a jeho subsystémov. Napríklad obsahuje aj vzdialenosti, ktoré boli manuálne namerané v teréne (viď. obrázok 6.2). Pomocou nich je možné vyhodnotiť aj samotnú presnosť automatickej kalibrácie. Taktiež je možné porovnať výsledky implementovaného systému s ostatnými systémami, ktorých výsledky tento dataset obsahuje.




Obr. 6.2: Príklad manuálne nameraných vzdialeností medzi značkami na rovine cesty. (Obr. prevzatý z [23])

## 6.2 Parametre testovania

Testovanie implementovaného systému prebiehalo na celej testovacej sade. Testované boli jednotlivé podsystémy (automatická kalibrácia, meranie rýchlosti), ich komponenty (získavanie úbežníkov, získavanie mierky) a aj kompletný systém ako celok. Zásadné parametre, za použitia ktorých bol podsystém automatickej kalibrácie testovaný, sú následovné:

- Systém bol spúšťaný s maskou prekrývajúcou okolie a pruhy smerujúce v protismere (použité boli masky obsiahnuté v datasete *BrnoCompSpeed*).
- Bol použitý detektor vozidiel s architektúrou *Faster R-CNN*, ktorý je priložený k implementovanému systému.
- Pri získavaní druhého úbežníku bol *Diamod Space* vymaskovaný na základe rozsahu ohniskovej vzdialenosti a uhlu natočenia horizontu z videí datasetu *BrnoCompSpeed*.
- Ako klasifikátor bola použitá sieť s architektúrou *ResNet50*. Použitý klasifikátor je taktiež priložený k tejto práci.

- Pri získavaní mierky bolo použitých sedem modelov vozidiel z modelovej rady značky *Škoda*, ktoré sa často vyskytujú na cestách v Českej republike (zoznam modelov sa nachádza v podkapitole 4.2.3).

Podsystém merania rýchlosti bol testovaný pri nasledujúcej konfigurácii:

- Systém bol spúšťaný bez masiek prekrývajúcich okolie a pruhy smerujúce v protismere.
- Stojace vozidlá boli odfiltrované.
- Ako kalibračné parametre boli použité v rôznych obmenách hodnoty získané implementovaným podsystémom automatickej kalibrácie a hodnoty získané manuálnou kalibráciou (jednotlivé kombinácie sú bližšie popísané v nasledujúcej kapitole týkajúcej sa výsledkov testovania, konkrétne v podkapitole 6.3.1).

## 6.3 Výsledky testovania

Hlavným výstupom testovania sú súbory generované podsystémom merania rýchlosti, ktoré obsahujú ako získané, resp. použité kalibračné parametre, tak údaje o prechádzajúcich vozidlách, ktoré systém zaznamenal. Ako už bolo spomenuté vyššie, systém bol testovaný vo viacerých konfiguráciách. Tieto konfigurácie sa líšia hlavne v tom, ako boli získané jednotlivé kalibračné parametre. Konkrétne verzie implementovaného systému, ktoré boli testované, a verzie existujúcich systémov, s ktorými budú výsledky testovania porovnávané, sú podrobnejšie popísané v nasledujúcej podkapitole.

### 6.3.1 Testované a porovnávané verzie

Verzie implementovaného systému, ktoré boli nad testovacou sadou spúšťané a testované, sú nasledujúce:

- **ManualCalib + ManualScale (MC + MS)** – V tejto verzii bol systém merania rýchlosti spúšťaný s kalibračnými údajmi získanými manuálnou kalibráciou. Jedná sa o údaje, ktoré sú k jednotlivým videám poskytnuté v rámci datasetu *BrnoCompSpeed*. Manuálne získané a zadané údaje predstavujú prvý a druhý úbežník spolu s mierkou scény.
- **AutomaticCalib + ManualScale (AC + MS)** – Pri tejto verzii bola zvolená konfigurácia sčasti pozostávajúca z údajov získaných implementovaným podsystémom automatickej kalibrácie a sčasti z manuálne získaných údajov. Automaticky bol získaný (pri použití nastavení popísaných v podkapitole 6.2) prvý a druhý úbežník, zatiaľ čo ako mierka scény bol použitý príslušný údaj z datasetu *BrnoCompSpeed*.
- **AutomaticCalib + AutomaticScale (AC + AS)** – Poslednou testovanou verziou implementovaného systému je konfigurácia, v ktorej sú všetky kalibračné parametre (oba úbežníky aj mierka scény) získané pomocou podsystému automatickej kalibrácie.

Verzie systémov, s ktorými boli výsledky testovania implementovaného systému porovnávané, sú tieto:

- **SochorCVIU ManualCalib + ManualScale (CVIU MC + MS)** – Tento systém bol navrhnutý a implementovaný Sochorom a kolektívom. Jeho kalibrácia vychádza z manuálne získaných vzdialeností zo snímanej situácie. Pre podrobnejší popis

určovania kalibračných parametrov z nameraných hodnôt medzi vyznačenými značkami, spôsob trackovania vozidel a merania ich rýchlosti viď. [22].

- **SochorCVIU Edgelets + BBScale (CVIU E + BBS)** – Tento porovnávaný systém bol taktiež navrhnutý a implementovaný Sochorom a kolektívom. Získavanie prvého a druhého úbežníku je založené na rovnakom princípe ako systém navrhnutý v tejto práci. Určovanie mierky je založené na napasovávaní vyrenderovaných modelov áut do ohraničujúceho boxu vozidla. Pre detailnejší popis viď. [22].

### 6.3.2 Vyhodnotenie výsledkov

Pri vyhodnocovaní výsledkov bol použitý evaluačný skript `eval.py`, ktorý je súčasťou datasetu *BrnoCompSpeed*. Ten umožňuje porovnávať výsledky na základe výsledkových súborov vo formáte `JSON`, ktorých štruktúra je spomínaná v kapitole 5. Skript ohodnocuje systémy z viacerých hľadísk.

#### Vyhodnotenie odhadu úbežníkov

Prvým hľadiskom, ktoré je vyhodnocované, je chyba v kalibrácii kamery. Presnejšie ide o chybu pri odhade úbežníkov, čiže nepresnosť ich umiestnenia. Vyhodnocovaný je rozdiel pomerov vzdialeností medzi značkami smerujúcimi k prvému úbežníku (červené čiary na obrázku 6.2) a vzdialeností medzi značkami smerujúcimi k druhému úbežníku (zelené čiary na obrázku 6.2). Nakoľko sú brané v úvahu len pomery vzdialeností (mierka scény nie je braná v úvahu), tak táto metrika uvažuje kalibráciu kamery iba vo forme dvoch detegovaných úbežníkov. V tabuľke 6.1 môžeme vidieť vyhodnotenie porovnávaných systémov a verzií systémov pre kalibráciu.

| systém | absolútna chyba | | | relatívna chyba [%] | | |
|---|---|---|---|---|---|---|
| | priemer | medián | 95 percentil | priemer | medián | 95 percentil |
| CVIU MC | 0,02 | 0,01 | 0,07 | 1,80 | 1,26 | 5,14 |
| CVIU E | 0,09 | 0,04 | 0,41 | 6,45 | 3,38 | 25,47 |
| AC | 0,11 | 0,04 | 0,40 | 7,60 | 4,32 | 27,18 |

Tabuľka 6.1: Chyby v pomeroch vzdialeností pre jednotlivé porovnávané spôsoby kalibrácie (*ManualCalib* implementovaného systému nie je uvádzaný, nakoľko sú úbežníky prevzaté zo systému *SochorCVIU ManualCalib*).

*AutomaticCalib*, čiže podsystém automatickej kalibrácie implementovaného systému, vykazuje mierne vyššiu priemernú absolútnu chybu v pomere vzdialeností ako porovnávaný systém *SochorCVIU Edgelets*. Medián a 95 percentil absolútnej chyby sú však rovnako dobré. V rámci relatívnej chyby je implementovaný systém horší v priemere o $1,15\%$, v mediáne chyby o $1,06\%$ a v 95 percentile chyby o $1,71\%$. Implementovaný systém sa teda presnosťou zisťovania úbežníkov približuje presnosti porovnávaného systému. Mierne vyššia nepresnosť je najpravdepodobnejšie spôsobená tým, že je systém orientovaný na rýchlosť získavania kalibračných parametrov. Konkrétne potreboval spracovať vo všetkých videách testovacej sady v priemere 3.865 snímok pre získanie prvého úbežníku a 1.122 snímok na získanie druhého úbežníku. Keďže systém spracováva každú druhú snímku a počet snímok

za sekundu vo videách datasetu je 50, tak na získanie prvého, resp. druhého úbežníku potrebuje spracovať len 2, 5 minúty, resp. 45 sekúnd videa.

### Vyhodnotenie merania vzdialenosti na rovine cesty

Druhým vyhodnocovaným hľadiskom je kalibrácia kamery spolu s mierkou scény. Na vyhodnocovanie sa znovu používajú manuálne získané vzdialenosti medzi značkami na rovine cesty. Prvé vyhodnotenie sa týka len vzdialeností medzi značkami smerujúcimi k prvému úbežníku (červené čiary na obrázku 6.2), keďže tento smer súhlasí so smerom pohybu prechádzajúcich áut a je tým pádom pri meraní rýchlosti najdôležitejší. Výsledky vyhodnotenia sú uvedené v tabuľke 6.2.

| systém | absolútna chyba [m] | | | relatívna chyba [%] | | |
|---|---|---|---|---|---|---|
| | priemer | medián | 95 percentil | priemer | medián | 95 percentil |
| CVIU MC + MS | 0,10 | 0,08 | 0,28 | 1,06 | 0,80 | 2,90 |
| CVIU E + BBS | 0,26 | 0,17 | 1,08 | 2,33 | 2,06 | 5,49 |
| AC + AS | 1,12 | 0,78 | 3,51 | 9,84 | 8,93 | 22,08 |
| AC + MS | 0,22 | 0,10 | 0,84 | 1,88 | 1,40 | 5,13 |

Tabuľka 6.2: Chyby porovnávaných systémov v nameraných vzdialenostiach na rovine cesty (iba v smere k prvému úbežníku).

Na základe výsledkov možno zhodnotiť, že implementovaný systém *AutomaticCalibration + AutomaticScale* nedosahuje v určovaní mierky scény výsledky porovnávaného systému *SochorCVIU Edgelets + BBScale*. Chyba v nameraných vzdialenostiach na rovine cesty (iba v smere k prvému úbežníku) je viac ako štyrikrát vyššia. V priemere to predstavuje odchylku o 1, 12 metra. Tieto nepresnosti sú spôsobené odchýlkami v získavaní úbežníkov a zlým spôsobom napasovávania modelu na detegované vozidlo. Zrotovaný model nie vždy zodpovedá vozidlu zo snímanej situácie (viď. obrázok 4.16 úplne vpravo) a do výpočtu vnáša chybu aj zväčšovanie modelu len na základe ohraničujúceho boxu, ten totiž nie vždy ohraničí úplne celé auto (viď. obrázok 4.3). Posledným faktorom, ktorý má vplyv na vysokú úroveň chyby, je, že sú pri výpočte použité body (stred kolesa), ktoré neležia priamo na rovine cesty. Výhodou implementovaného systému je, že na získanie mierky potrebuje spracovať v priemere iba 3.048 snímok, čiže približne dve minúty videa.

Pokiaľ je ale použitá manuálne získaná mierka (*AutomaticCalib + ManualScale*), tak systém dosahuje dobré výsledky. Samotná automatická kalibrácia v podobe získania prvých dvoch úbežníkov môže teda výrazne znížiť náročnosť práce, ktorú musí užívateľ pri použití systému vynaložiť, a zároveň zachováva presnosť v dobrom rozsahu.

V tabuľke 6.3 môžeme vidieť vyhodnotenie chýb v nameraných vzdialenostiach medzi všetkými značkami na rovine cesty, nielen tými, čo smerujú k prvému úbežníku. Trend, ktorý je zaznamenaný v tabuľke 6.2, je zachovaný. Implementovaný systém s automatickou kalibráciou (*AC + AS*) má zvýšenú nepresnosť, avšak, pokiaľ sú brané do úvahy chyby vo všetkých nameraných vzdialenostiach (nie len smerom k prvému úbežníku), tak je absolútna chyba nižšia v priemere o 0, 1 metra a dosahuje hodnty len trojnásobku chyby porovnávaného systému *CVIU E + BBS*. Na rozdiel od *AC + AS*, verzia s manuálnym získavaním mierky scény (*AC + MS*) a verzia *CVIU E + BBS* porovnávaného systému majú vyš-

| systém | absolútna chyba [m] | | | relatívna chyba [%] | | |
|---|---|---|---|---|---|---|
| | priemer | medián | 95 percentil | priemer | medián | 95 percentil |
| CVIU MC + MS | 0,08 | 0,05 | 0,24 | 1,03 | 0,58 | 3,22 |
| CVIU E + BBS | 0,34 | 0,18 | 2,29 | 3,47 | 2,28 | 30,49 |
| AC + AS | 1,02 | 0,69 | 3,44 | 10,73 | 9,79 | 22,24 |
| AC + MS | 0,48 | 0,17 | 1,58 | 5,59 | 2,20 | 21,61 |

Tabuľka 6.3: Chyby porovnávaných systémov v nameraných vzdialenostiach na rovine cesty (*MC + MS* implementovaného systému nie je uvádzaný, nakoľko sú kalibračné parametre prevzaté zo systému *SochorCVIU MC + MS*).

šiu chybu vo všetkých nameraných vzdialenostiach na rovine cesty ako pri vzdialenostiach medzi značkami smerujúcimi k prvému úbežníku.

### Vyhodnotenie merania rýchlosti

Posledným a najvýznamnejším hodnotiacim faktorom je presnosť merania rýchlosti. Chybovosť jednotlivých porovnávaných systémov a verzií systémov oproti hodnotám nameraným pomocou optických brán za použitia LIDAR-u pri zhotovovaní datasetu BrnoCompSpeed, je možné vidieť v tabuľke 6.4.

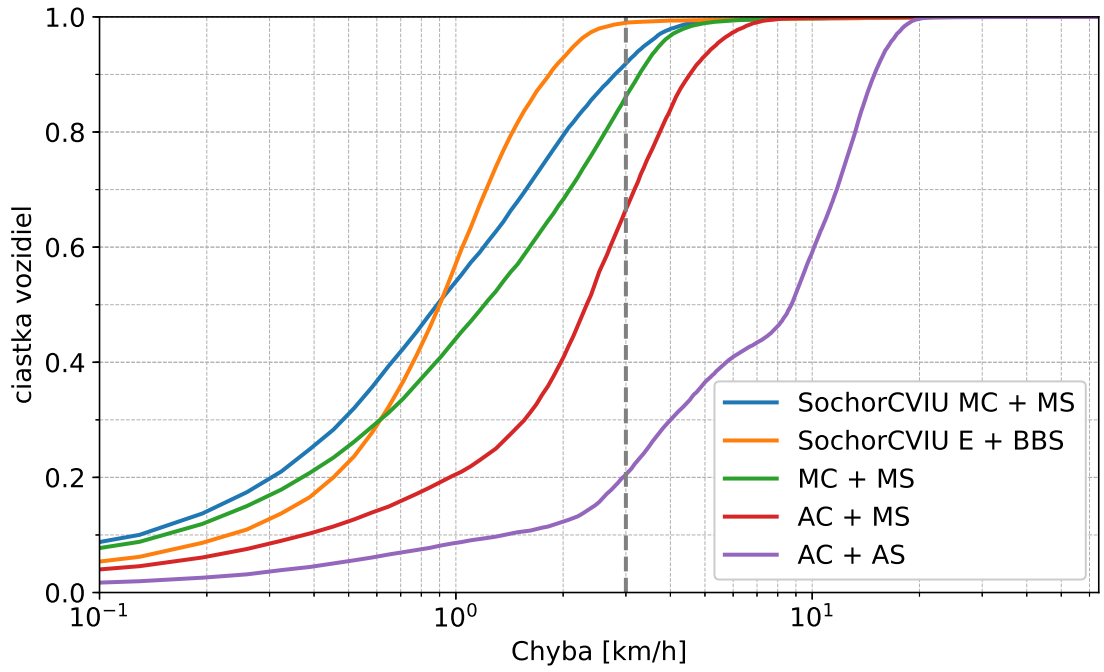
| systém | absolútna chyba [km/h] | | | relatívna chyba [%] | | |
|---|---|---|---|---|---|---|
| | priemer | medián | 95 percentil | priemer | medián | 95 percentil |
| CVIU MC + MS | 1,21 | 0,91 | 3,17 | 1,51 | 1,15 | 3,94 |
| CVIU E + BBS | 1,10 | 0,97 | 3,05 | 1,39 | 1,22 | 4,13 |
| MC + MS | 1,62 | 1,39 | 3,69 | 2,02 | 1,78 | 4,59 |
| AC + AS | 8,15 | 7,98 | 17,74 | 10,15 | 11,18 | 21,40 |
| AC + MS | 2,45 | 2,12 | 5,72 | 3,03 | 2,68 | 6,66 |

Tabuľka 6.4: Chyby porovnávaných systémov v nameraných rýchlostiach prechádzajúcich áut.

Z týchto výsledkov je možné vyhodnotiť presnosť samotného podsystému merania rýchlosti. *ManualCalib + ManualScale*, predstavujúci podsystém merania rýchlosti navrhnutý a implementovaný v rámci tejto práce, je v priemere len o $0,41$ km/h nepresnejší oproti porovnávanému systému *SochorCVIU ManualCalib + ManualScale* z práce [22] (oba systémy boli vyhodnotené s totožnými kalibračnými parametrami). Mierne zvýšená nepresnosť implementovaného systému môže byť spôsobená určovaním referenčného bodu sledovaného vozidla v jednotlivých snímkach. Kým v systéme tejto práce je použitý stred spodnej hrany 2D ohraničujúceho boxu, v práci od Sochora a kolektívu je ako referenčný bod určený stred prednej spodnej hrany 3D ohraničujúceho boxu. Presnosť je teda znížená v prospech jednoduchšieho výpočtu (nie je potrebné zostavovať 3D ohraničujúce boxy pre každé vozidlo).

Ďalej je možné vyhodnotiť, že pokiaľ je spolu s automatickou kalibráciou použitá manuálne získaná mierka (systém *AutomaticCalibration + ManualScale*), tak je priemerná absolútna chyba v meraní rýchlosti vyššia len o $1,34$ km/h oproti *CVIU MC + MS* a o $0,83$ km/h oproti *MC + MS*. K chybe, ktorú zavádza do výsledkov systém merania rýchlosti, je teda pridaná ešte aj chyba automatickej kalibrácie. Priemerná chyba $2,45$ km/h je však stále prijateľná, a teda systém dokáže relatívne presne merať rýchlosť prechádzajúcich atomobilov len s jedným manuálne získaným vstupným parametrom.

Avšak chyba, ktorá vznikla pri automatickom získavaní mierky, sa odrazila na chybe v meraní rýchlosti. Systém *AutomaticCalib + AutomaticScale* tak dosahuje absolútnu chybu v meraní v priemere až $8,15$ km/h. Na základe tohto údaju je možné zhodnotiť, že nepresnosť implementovaného podsystému na získavanie mierky má fatálne následky na presnosť celého automatického systému. Kumulatívny histogram absolútnych chýb pre jednotlivé porovnávané systémy môžeme vidieť na obrázku 6.3. Systém automatickej kalibrácie tak splnil cieľ v rámci optimalizácie na čas potrebný pre získanie kalibračných parametrov (najviac potrebuje spracovať v priemere dve a pol minúty videa pre získanie prvého úbežníku), avšak jeho presnosť nedosahuje úroveň presnosti niektorých už existujúcich riešení.




Obr. 6.3: Kumulatívny histogram absolútnych chýb jednotlivých verzií implementovaného systému a porovnávaných systémov.

# Kapitola 7

# Záver

Cieľom tejto diplomovej práce bolo navrhnúť a naimplementovať systém pre kalibráciu a meranie rýchlosti automobilov prechádzajúcich pred dohľadovou kamerou. Pred samotným návrhom systému boli preštudované algoritmy pre automatickú kalibráciu dopravnej dohľadovej kamery a algoritmy pre meranie rýchlosti pomocou jednej nakalibrovanej dohľadovej kamery. Na základe získaných vedomostí bol vypracovaný návrh rozsiahleho systému automatickej kalibrácie a merania rýchlosti, ktorý bol následne úspešne naimplementovaný.

Systém bol navrhnutý a naimplementovaný ako uzatvorený celok. Nie je teda potrebné spúšťať rôzne skripty, vyhodnocovať medzivýsledky a podobne. Vďaka tomu je jednoduchší na použitie a jeho používateľovi stačí systém spustiť s požadovanými parametrami a zdrojmi a už len čakať na výsledky. Podsystém vykonávajúci automatickú kalibráciu bol navrhnutý a naimplementovaný tak, aby potreboval spracovať čo najkratší časový úsek videa. V priemere tak podsystém potrebuje pri automatickej kalibrácii načítať a spracovať len dva a pol minúty vstupného videa. Tieto údaje boli získané pri jeho testovaní a vyhodnocovaní nad datasetom *BrnoCompSpeed*.

Testovaná a vyhodnocovaná bola aj presnosť celého systému a jeho podsystémov. Kým presnosť podsystému na meranie rýchlosti je v porovnaní s predchádzajúcimi systémami, vzhľadom na zjednodušenie výpočtu, dostačujúca (chyba v meraní rýchlosti je pri manuálnom získaní kalibračných parametrov v priemere len $1,62$ km/h), podsystém automatickej kalibrácie dosahuje horšie výsledky. Pokiaľ sú pri meraní použité kalibračné parametre získané automatickou kalibráciou, tak je chyba v meraní rýchlosti v priemere $8,15$ km/h. Veľkosť tejto chyby je spôsobená nepresným určovaním mierky scény. Pokiaľ je nahradená manuálne získaným údajom, tak je priemerná chyba iba $2,45$ km/h. Dá sa teda zhrnúť, že výsledný naimplementovaný systém, s výnimkou podsystému na určovanie mierky scény, dosahuje uspokojivé výsledky.

V budúcnosti by bolo potrebné zamerať sa práve na zlepšenie získavania mierky scény. To by sa dalo dosiahnuť zvýšením zložitosti jej výpočtu. Napríklad by bolo možné zlepšiť spôsob, akým je model reálneho vozidla natáčaný tak, aby korešpondoval s detegovaným vozidlom. K zlepšeniu presnosti by taktiež mohlo viesť zavedenie ďalšieho spracovania pri samotnej detekcii vozidiel v obraze. Ohraničujúce boxy totiž nie vždy obsahujú celé vozidlo a aj z toho dôvodu nezodpovedá model detegovanému vozidlu s potrebnou presnosťou. Ďalšiu možnosť zvýšenia presnosti predstavuje projekcia bodov, z ktorých sa získava vzdialenosť na vozidlách, na rovinu cesty.

# Literatúra


[1] Convolutional Neural Network [online].
URL https://ch.mathworks.com/fr/discovery/convolutional-neural-network.html

[2] Keras [online].
URL https://www.keras.io/

[3] TensorFlow [online].
URL https://www.tensorflow.org/

[4] Chaperon, T.; Droulez, J.; Thibault, G.: Reliable camera pose and calibration from a small set of point and line correspondences: A probabilistic approach. *Computer Vision and Image Understanding*, ročník 115, č. 5, 2011: s. 576–585, ISSN 1077-3142.

[5] Dailey, D. J.; Cathey, F. W.; Pumrin, S.: An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, ročník 1, č. 2, Jun 2000: s. 98–107, ISSN 1524-9050.

[6] Dawson, D. N.; Birchfield, S. T.: An Energy Minimization Approach to Automatic Traffic Camera Calibration. *IEEE Transactions on Intelligent Transportation Systems*, ročník 14, č. 3, Sept 2013: s. 1095–1108, ISSN 1524-9050.

[7] Do, V. H.; Nghiem, L. H.; Thi, N. P.; aj.: A simple camera calibration method for vehicle velocity estimation. In *2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, June 2015, s. 1–5.

[8] Dubská, M.; Herout, A.: Real Projective Plane Mapping for Detection of Orthogonal Vanishing Points. In *Proceedings of BMVC 2013*, 2013, s. 90.1–90.10.

[9] Dubska, M.; Herout, A.; Juranek, R.; aj.: Fully Automatic Roadside Camera Calibration for Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*, ročník 16, č. 3, June 2015: s. 1162–1171, ISSN 1524-9050.

[10] Dubská, M.; Sochor, J.; Herout, A.: Automatic Camera Calibration for Traffic Understanding. In *Proceedings of BMVC 2014*, 2014, s. 1–10.

[11] George Shiu Kai Fung, G. K. P., Nelson Hon Ching Yung: Camera calibration from road lane markings. *Optical Engineering*, ročník 42, 2003: s. 42–42–11.

[12] Grammatikopoulos, L.; Karras, G.; Petsa, E.: Automatic estimation of vehicle speed from uncalibrated video sequences. In *International Synopsium on Modern*

*Technologies, Education and Professional Practice in Geodesy and Related Fields*, Nov 2005, s. 332–338.

[13] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. *CoRR*, ročník abs/1512.03385, 2015.

[14] He, X. C.; Yung, N. H. C.: A Novel Algorithm for Estimating Vehicle Speed from Two Consecutive Images. In *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, Feb 2007, ISSN 1550-5790, s. 12–12.

[15] Huang, J.; Rathod, V.; Sun, C.; aj.: Speed/accuracy trade-offs for modern convolutional object detectors. 2016.

[16] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. 2015.

[17] Luvizon, D. C.; Nassu, B. T.; Minetto, R.: Vehicle speed estimation by license plate detection and tracking. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, ISSN 1520-6149, s. 6563–6567.

[18] Luvizon, D. C.; Nassu, B. T.; Minetto, R.: A Video-Based System for Vehicle Speed Measurement in Urban Roadways. *IEEE Transactions on Intelligent Transportation Systems*, ročník 18, č. 6, June 2017: s. 1393–1404, ISSN 1524-9050.

[19] Pai, T.-W.; Juang, W.-J.; Wang, L.-J.: An adaptive windowing prediction algorithm for vehicle speed estimation. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, 2001, s. 901–906.

[20] Ren, S.; He, K.; Girshick, R.; aj.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, č. 6, June 2017: s. 1137–1149, ISSN 0162-8828.

[21] Schoepflin, T. N.; Dailey, D. J.: Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, ročník 4, č. 2, June 2003: s. 90–98, ISSN 1524-9050.

[22] Sochor, J.; Juranek, R.; Herout, A.: Traffic surveillance camera calibration by 3D model bounding box alignment for accurate vehicle speed measurement. *Computer Vision and Image Understanding*, ročník 161, 2017: s. 87–98, ISSN 1077-3142.

[23] Sochor, J.; Juránek, R.; Špaňhel, J.; aj.: BrnoCompSpeed: Review of Traffic Camera Calibration and Comprehensive Dataset for Monocular Speed Measurement. 2017.

[24] Sochor, J.; Špaňhel, J.; Herout, A.: BoxCars: Improving Fine-Grained Recognition of Vehicles Using 3-D Bounding Boxes in Traffic Surveillance. *IEEE Transactions on Intelligent Transportation Systems*, ročník PP, č. 99, 2018: s. 1–12, ISSN 1524-9050.

[25] Song, K. T.; Tai, J. C.: Dynamic Calibration of Pan-Tilt-Zoom Cameras for Traffic Monitoring. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, ročník 36, č. 5, Oct 2006: s. 1091–1103, ISSN 1083-4419.

[26] Wang, K.; Huang, H.; Li, Y.; aj.: Research on lane-marking line based camera calibration. *2007 IEEE International Conference on Vehicular Electronics and Safety*, Dec 2007: s. 1–6.

[27] Wang, Z.; Tang, L.; Liu, X.; aj.: Orientation Invariant Feature Embedding and Spatial Temporal Regularization for Vehicle Re-Identification. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[28] You, X.; Zheng, Y.: An accurate and practical calibration method for roadside camera using two vanishing points. *Neurocomputing*, ročník 204, č. Supplement C, 2016: s. 222–230, ISSN 0925-2312.

# Príloha A

# Obsah priloženého pamäťového média

- `doc/` – zdrojové súbory technickej správy
- `results/` – výsledkové súbory pre dataset *BrnoCompSpeed*
- `src/` – zdrojové súbory implementovaného systému
- `video/` – video pre prezentáciu projektu
- `README.md` – manuál k priloženému systému
- `xjaklo02-Mereni-rychlosti-automobilu-z-dohledove-kamery.pdf` – technická správa

# Príloha B

# Spustenie

Pred samotným spustením systému je potrebné spustiť `Makefile`, ktorý spúšťa skript `setup.py`. Ten má za úlohu preložiť a do obálky pre `Python` zabaliť `accumulator.cpp`, aby ho mohla využívať trieda slúžiaca ako *Diamond Space Accumulator*. Celý implementovaný systém slúžiaci na meranie rýchlosti automobilov z dohľadovej kamery sa následne spúšťa pomocou skriptu `speed_measurement_system.py`. Ten sa spúšťa nasledovne:

```
speed_measurement_system.py [-h] [--mask_source MASK_SOURCE]
            [--vp1 VP1 VP1] [--vp2 VP2 VP2] [--scale SCALE]
            video_source detector_source classificator_source
            run_calibration use_mask filter_standing_vehicles
            print_speed cache_frames fps
```

Jednotlivé pozičné parametre majú nasledujúci význam:

- `video_source` – *reťazec* predstavujúci cestu k súboru s videom, nad ktorým má prebehnúť kalibrácia a/alebo meranie rýchlosti,
- `detector_source` – *reťazec* predstavujúci cestu k priečinku, v ktorom sa nachádza súbor `frozen_inference_graph.pb` s predtrénovaným detektorom,
- `classificator_source` – *reťazec* predstavujúci cestu k súboru s predtrénovaným klasifikátorom uloženým vo formáte `H5`,
- `run_calibration` – *boolean* predstavujúci príznak, či má byť nad vstupným videom spustená aj kalibrácia alebo iba meranie rýchlosti (`True` – aj kalibrácia, `False` – iba meranie),
- `use_mask` – *boolean* predstavujúci príznak, či má byť pri kalibrácii a meraní použitá maska prekrývajúca určité plochy videa,
- `filter_standing_vehicles` – *boolean* predstavujúci príznak, či majú byť odfiltrované nepohybujúce sa vozidlá zo záznamu,
- `print_speed` – *boolean* predstavujúci príznak, či má byť do výstupného súboru vo formáte `JSON` priradená ku každému detegovanému autu aj informácia o jeho rýchlosti v kilometroch za hodinu,
- `cache_frames` – *boolean* predstavujúci príznak, či majú byť ukladané snímky s ohraničujúcimi boxami, spracované podsystémom automatickej kalibrácie, pre potreby podsystému merania rýchlosti,

- `fps` – *float* reprezentujúci počet snímok za sekundu vo vstupnom videu.

Okrem pozičných parametrov je možné pri spustení použiť aj voliteľný parameter:

- `-h, --help` – vypíše nápovedu a ukončí program.

Ďalší voliteľný parameter je viazaný na nastavenie parametra `use_mask`. Pokiaľ je nastavený na `True`, tak je potrebné použiť aj nasledovný parameter:

- `--mask_source MASK_SOURCE` – *reťazec* predstavujúci cestu k čiernobielemu obrázku slúžiacemu ako maska, ktorá sa má pri spracovávaní vstupného videa použiť (oblasti pokryté čiernou farbou sú vynechané).

Ostatné voliteľné parametre sú potrebné v prípade, že je nastavený pozičný parameter `run_calibration` na `False`, a teda je spúšťaný len systém merania rýchlosti. Ich význam je nasledovný:

- `--vp1 VP1 VP1` – dva parametre typu *float* predstavujúce x-ovú a y-ovú súradnicu bodu v obraze, pričom tento bod zodpovedá prvému úbežníku scény vstupného videa,
- `--vp2 VP2 VP2` – dva parametre typu *float* predstavujúce x-ovú a y-ovú súradnicu bodu v obraze, pričom tento bod zodpovedá druhému úbežníku scény vstupného videa,
- `--scale SCALE` – jeden parameter typu *float* predstavujúci mierku scény vstupného videa.

Systém je možné kedykoľvek počas prebiehajúceho výpočtu ukončiť stlačením klávesy Esc v okne s demonštatívnym grafickým výstupom.

**Príklad spustenia č. 1:**

```
speed_measurement_system.py data/session1_left/video.avi
            detector/vehicle_frcnn classifier/ResNet50.h5
            True True True False True 50.0
            --mask_source data/session1_left/video_mask.png
```

Uvedený príkaz spustí systém merania rýchlosti aj s kalibráciou nad `video.avi` s využitím detektora v priečinku `vehicle_frcnn` a klasifikátora `ResNet50.h5`. Ako masku použije `video_mask.png` a hodnota počtu snímok za sekundu je 50. Stojace vozidlá budú odfiltrované a do výstupného súboru nebudú vpisované rýchlosti. *Cacheovanie* snímok je zapnuté.

**Príklad spustenia č. 2:**

```
speed_measurement_system.py data/session2_right/video.avi
            detector/vehicle_frcnn_res101 classifier/ResNet50.h5
            False False True True False 50.0
            --vp1 -389.868995633 -474.497816594
            --vp2 4604.081632653061 -570.2040816326527
            --scale 0.0219
```

Uvedený príkaz spustí systém merania rýchlosti bez kalibrácie nad `video.avi` s využitím detektora v priečinku `vehicle_frcnn_res101` a klasifikátora `ResNet50.h5`. Systém pobeží bez využitia masky a hodnota počtu snímok za sekundu je 50. Stojace vozidlá budú odfiltrované a do výstupného súboru budú vpísané aj rýchlosti. Ako kalibračné parametre budú použité prvý úbežník ležiaci v bode $\mathbf{u} = [-389.868995633, -474.497816594]$, druhý úbežník ležiaci v bode $\mathbf{v} = [4604.081632653061, -570.2040816326527]$ a mierka scény $\lambda = 0.0219$. *Cacheovanie* snímok je vypnuté.

# Príloha C

# Systémové požiadavky

Pre správny beh implementovaného systému je potrebné splniť nasledujúce požiadavky. Najzákladnejšou podmienkou pre spustenie je nainštalovaný:

- `Python` 3.6.

Taktiež je potrebné, aby okrem základných knižníc boli k nemu doinštalované nasledujúce knižnice:

- `OpenCV`,
- `NumPy`,
- `SciPy`,
- `Cython`,
- `Psutil`,
- `Keras`,
- `Tensorflow`.

Posledný framework (`Tensorflow`) so sebou prináša ďalšie systémové požiadavky, ktoré je potrebné splniť, aby bol funkčný. Konkrétne sa jedná o požiadavky na operačný systém (založený na 64-bit architektúre):

- *macOS* 10.12.6 (*Sierra*) alebo novší,
- *Ubuntu* 16.04 alebo novší,
- *Windows* 7 alebo novší.

`Tensorflow` môže bežať buď na CPU alebo GPU. Pre spustenie a beh implementovaného systému je ale vysoko odporúčané, vzhľadom na jeho výpočetnú náročnosť, použiť verziu pre GPU. Jej požiadavky sú nasledovné:

- *Cuda Toolkit* 9.0,
- *NVIDIA* ovládače spojené s *Cuda Toolkit* 9.0,
- *cuDNN* v7.0,

- GPU s *CUDA Compute Capability* 3.0 alebo vyššiou pre preloženie zo zdroja a 3.5 alebo vyššiou pre použitie binárnych súborov. Zoznam podporovaných GPU možno nájsť v dokumentácii od *NVIDIA*.

Okrem GPU má systém hardvérové požiadavky aj na pamäť RAM. Pre úspešné ukončenie výpočtu systému je potrebné mať aspoň:

- 32GB RAM pre beh celého systému aj s automatickou kalibráciou,
- 8GB RAM pre beh samotného podsystému merania rýchlosti.